FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Interactive Visual Analysis in Automotive Engineering Design

DISSERTATION

zur Erlangung des akademischen Grades

Doktor/in der technischen Wissenschaften

eingereicht von

Zoltán Konyha, MSc.
Matrikelnummer 0627536

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Priv.-Doz. Dipl.-Ing. Dr.techn. Helwig Hauser

Diese Dissertation haben begutachtet:

(Priv.-Doz. Dipl.-Ing. Dr.techn. Helwig Hauser)	(Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller)

Wien, 19.12.2012

(Zoltán Konyha, MSc.)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Interactive Visual Analysis in Automotive Engineering Design

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor/in der technischen Wissenschaften

by

Zoltán Konyha, MSc.
Registration Number 0627536

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Priv.-Doz. Dipl.-Ing. Dr.techn. Helwig Hauser

The dissertation has been reviewed by:

(Priv.-Doz. Dipl.-Ing. Dr.techn. Helwig Hauser)	(Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller)

Wien, 19.12.2012

(Zoltán Konyha, MSc.)

Erklärung zur Verfassung der Arbeit

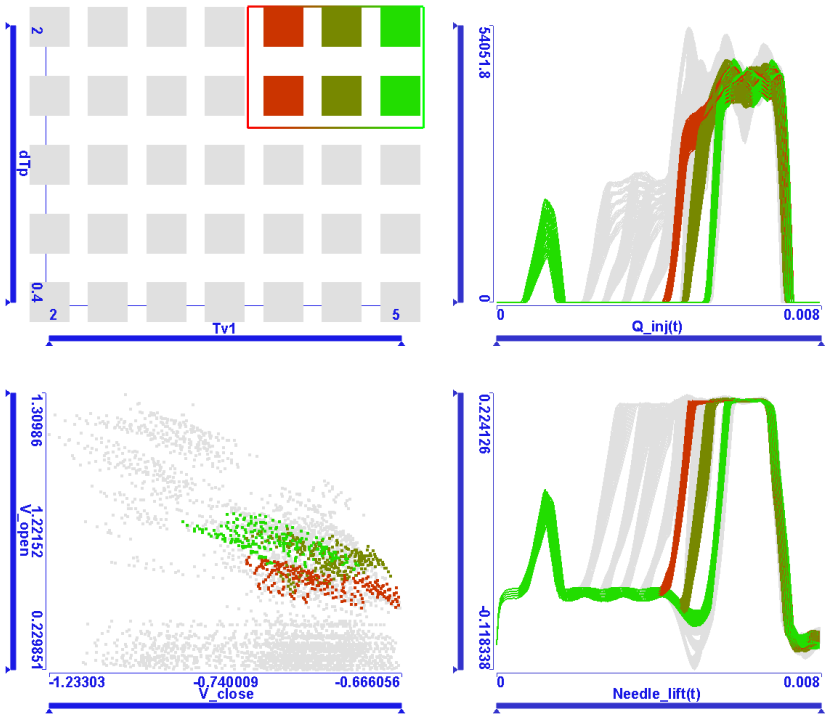
Zoltán Konyha, MSc.
Lilienthalgasse 39, 8020 Graz

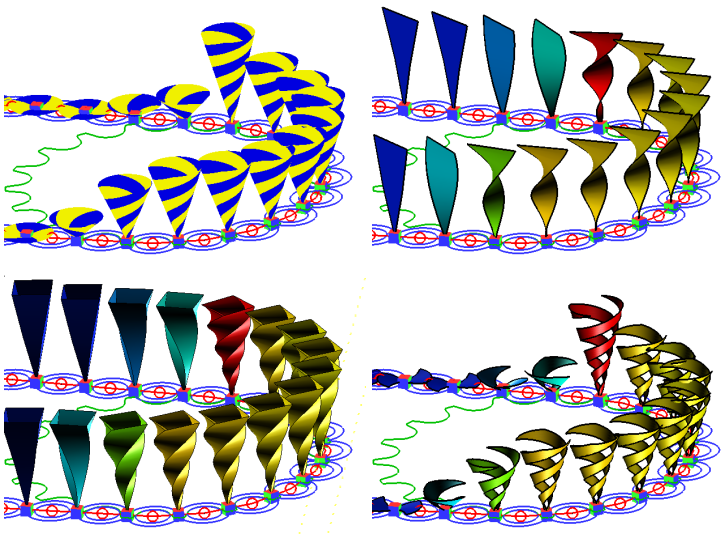
Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Zoltán Konyha, MSc.)

Interactive Visual Analysis in Automotive Engineering Design

Zoltán Konyha, PhD thesis





mailto:Konyha@VRVis.at

To my wife Edit,
and to my daughters
Anna and Julia

Abstract

Computational simulation has become instrumental in the design process in automotive engineering. Virtually all components and subsystems of automobiles can be simulated. The simulation can be repeated many times with varied parameter settings, thereby simulating many possible design choices. Each simulation run can produce a complex, multivariate, and usually time-dependent result data set. The engineers' goal is to generate useful knowledge from those data. They need to understand the system's behavior, find correlations in the results, conclude how results depend on the parameters, find optimal parameter combinations, and exclude the ones that lead to undesired results.

Computational analysis methods are widely used and necessary to analyze simulation data sets, but they are not always sufficient. They typically require that problems and interesting data features can be precisely defined from the beginning. The results of automated analysis of complex problems may be difficult to interpret. Exploring trends, patterns, relations, and dependencies in time-dependent data through statistical aggregates is not always intuitive.

In this thesis, we propose techniques and methods for the interactive visual analysis (IVA) of simulation data sets. Compared to computational methods, IVA offers new and different analysis opportunities. Visual analysis utilizes human cognition and creativity, and can also incorporate the experts' domain knowledge. Therefore, their insight into the data can be amplified, and also less precisely defined problems can be solved.

We introduce a data model that effectively represents the multi-run, time-dependent simulation results as *families of function graphs*. This concept is central to the thesis, and many of the innovations in this thesis are closely related to it. We present visualization techniques for families of function graphs. Those visualizations, as well as well-known information visualization plots, are integrated into a coordinated multiple views framework. All views provide focus+context visualization. Compositions of brushes spanning several views can be defined iteratively to select interesting features and promote information drill-down. Valuable insight into the spatial aspect of the data can be gained from (generally domain-specific) spatio-temporal visualizations. In this thesis, we propose interactive, glyph-based 3D visualization techniques for the analysis of rigid and elastic multibody system simulations.

We integrate the on-demand computation of derived data attributes of families of function graphs into the analysis workflow. This facilitates the selection of deeply hidden data features that cannot be specified by combinations of simple brushes on the original data attributes. The combination of these building blocks supports interactive knowledge discovery. The analyst can build a mental model of the system; explore also unexpected features and relations; and gene-

rate, verify or reject hypotheses with visual tools; thereby gaining more insight into the data. Complex tasks, such as parameter sensitivity analysis and optimization can be solved. Although the primary motivation for our work was the analysis of simulation data sets in automotive engineering, we learned that this data model and the analysis procedures we identified are also applicable to several other problem domains. We discuss common tasks in the analysis of data containing families of function graphs.

Two case studies demonstrate that the proposed approach is indeed applicable to the analysis of simulation data sets in automotive engineering. Some of the contributions of this thesis have been integrated into a commercially distributed software suite for engineers. This suggests that their impact can extend beyond the visualization research community.

Kurzfassung

Computersimulationen spielen für Designprozesse in der Automobilindustrie eine entscheidende Rolle. So gut wie alle Komponenten und Subsysteme von Autos können simuliert werden. Da Simulationen mit verschiedenen Parameterkombinationen immer wieder aufs Neue durchgeführt werden können, ergeben sich viele verschiedene Designmöglichkeiten. Jeder Simulationsdurchgang kann einen komplexen, multivariaten und normalerweise zeitabhängigen Ergebnisdatensatz erzeugen. Das Ziel der Ingenieure ist es, aus diesen Daten nutzbringendes Wissen zu generieren. Dazu müssen sie das Verhalten des Systems verstehen, optimale Parameterkombinationen finden und jene Kombinationen ausschließen, die zu unerwünschten Resultaten führen.

Automatische Datenanalysemethoden sind weit verbreitet und auch notwendig, um Simulationsergebnisse zu analysieren, aber sie sind nicht immer ausreichend. Sie setzen typischerweise voraus, dass die Problemstellung und interessante Datenmerkmale bereits am Anfang präzise definierbar sind. Die Ergebnisse einer automatischen Analyse von komplexen Problemstellungen können jedoch schwierig zu interpretieren sein. Das Erforschen von Trends, Mustern, Relationen und Abhängigkeiten in zeitabhängigen Datensätzen durch statistische Merkmale ist nicht immer einfach und unmittelbar zugänglich.

In dieser Dissertation schlagen wir Techniken und Methoden für die interaktive visuelle Analyse (IVA) von Simulationsergebnissen vor. Im Vergleich zu automatischen Methoden bietet IVA neue und andersartige Analysemöglichkeiten. Die visuelle Analyse stützt sich auf die menschliche Wahrnehmung und Kreativität und kann zusätzlich das Fachwissen der Ingenieure integrieren. Auf diese Weise kann das Verständnis der Daten vertieft werden und auch weniger exakt definierte Problemstellungen können gelöst werden.

Wir stellen ein Datenmodell vor, das die mehrlagigen, zeitabhängigen Simulationsresultate als Funktionenschar darstellt. Dieses Konzept steht im Zentrum dieser Dissertation und viele darin präsentierte Innovationen stehen mit ihm in engem Zusammenhang. Wir präsentieren Visualisierungstechniken für Funktionenscharen. Genauso wie bekannte Informationsvisualisierungsdarstellungen sind diese Visualisierungen in ein koordiniertes Mehrbildsystem eingebettet. Alle Ansichten stellen Fokus+Kontext-Visualisierungen zur Verfügung.

Markierungen können über mehreren Ansichten iterativ kombiniert werden, um interessante Merkmale zu definieren und um das Informationssuchen zu erleichtern. Wertvolle Einsichten bezüglich des räumlichen Aspekts der Daten können durch (üblicherweise fachspezifische) Raum-Zeit-Visualisierungen erreicht werden. In dieser Dissertation schlagen wir außerdem interaktive, glyph-basierte 3D-Visualisierungstechniken für die Analyse von starren und elastischen Mehr-körpersystemen vor.

Wir integrieren bedarfsbasierte Berechnungen von abgeleiteten Datenattributen von Funktionenscharen in den Analyseablauf. Dies ermöglicht das Aufspüren von versteckten Datenmerkmalen, die durch Kombinationen von einfachen Markierungen auf den ursprünglichen Datenattributen nicht spezifiziert werden können. Die Kombination dieser Komponenten unterstützt das interaktive Auffinden von Wissen. Die Analytiker können ein gedankliches Modell des Systems konstruieren, können auch unerwartete Merkmale und Beziehungen erforschen, können Hypothesen über visuelle Tools generieren, verifizieren oder verwerfen, und können dadurch einen Erkenntnisgewinn über die Daten erlangen. Komplexe Aufgaben, wie zum Beispiel eine Parametersensitivitätsanalyse oder Optimierungen, können gelöst werden. Obwohl die ursprüngliche Motivation für unsere Arbeit die Analyse von Simulationsdaten in der Automobilindustrie war, zeigte sich, dass dieses Datenmodell und die Analyseverfahren, die wir identifiziert haben, auch auf viele andere Problemfelder anwendbar sind. Wir diskutieren daher übliche Aufgaben in der Analyse von Daten, die Funktionenscharen enthalten.

Zwei Fallstudien zeigen, dass der vorgeschlagene Ansatz tatsächlich auf die Analyse von Simulationsergebnissen in der Automobilindustrie anwendbar ist. Einige der Beiträge dieser Dissertation wurden bereits in eine kommerziell verbreitete Software für Ingenieure integriert, was darauf hindeutet, dass ihre Wirkung weit über die Kreise der Visualisierungsforschung hinausgehen kann.

Related Publications

This thesis is based on the following publications:

Zoltán Konyha, Krešimir Matković, and Helwig Hauser
Interactive 3D Visualization Of Rigid Body Systems
Proceedings of the IEEE *Visualization (VIS 2003)*, pages 539-546, 2003.

Zoltán Konyha, Josip Jurić, Krešimir Matković, and Jürgen Krasser
Visualization of Elastic Body Dynamics for Automotive Engine Simulations
Proceedings of the IASTED *Visualization, Imaging, and Image Processing (VIIP 2004)*, pages 742–747, 2004

Zoltán Konyha, Krešimir Matković, Denis Gračanin, Mario Jelović, and Helwig Hauser
Interactive Visual Analysis of Families of Function Graphs
IEEE *Transactions on Visualization and Computer Graphics*, 12(6), pages 1373-1385, 2006.

Zoltán Konyha, Krešimir Matković, Denis Gračanin, and Mario Duras
Interactive Visual Analysis of a Timing Chain Drive Using Segmented Curve View and other Coordinated Views
Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007), pages 3-15, 2007

Krešimir Matković, Denis Gračanin, Zoltán Konyha, and Helwig Hauser
Color Lines View: An Approach to Visualization of Families of Function Graphs
Proceedings of the 11th International Conference Information Visualization (IV'07), pages 59-64, 2007.

Zoltán Konyha, Alan Lež, Krešimir Matković, Mario Jelović, and Helwig Hauser
Interactive Visual Analysis of Families of Curves using Data Aggregation and Derivation
Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies (i-KNOW '12), pages 31–38, 2012.

The following publications are also related to this thesis:

Krešimir Matković, Josip Jurić, Zoltán Konyha, Jürgen Krasser, and Helwig Hauser
Interactive Visual Analysis of Multi-Parameter Families of Function Graphs
Proceedings of the Third International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2005), pages 54-62, 2005

Krešimir Matković, Mario Jelović, Josip Jurić, Zoltán Konyha, and Denis Gračanin
Interactive Visual Analysis and Exploration of Injection Systems Simulations
Proceedings of the IEEE Visualization (VIS 2005), pages 391-398, 2005.

Zoltán Konyha, Krešimir Matković, and Helwig Hauser
Interactive Visual Analysis in Engineering: A Survey
Posters at the 25th Spring Conference on Computer Graphics (SCCG 2009), pages 31–38, 2009.

Contents

Chapter 1

Introduction and Overview

"The soul never thinks without a picture."

— Aristotle (384–322 BC)[1]

This thesis presents new methods for the interactive visual analysis of automotive engineering simulation data sets. Visual analysis can amplify the engineers' cognition of simulation results and facilitates the generation of useful knowledge from raw data attributes. Understanding the complex relationships in the data helps engineers perform typical design tasks and solve common problems while designing complex subsystems found in modern automobiles. In this chapter, we first provide a brief description of the problem domain, the design process in automotive engineering. Then a short introduction to the proposed methodology, interactive visual analysis, is given. Finally, we outline the main contributions of this work and provide an overview of the structure of this thesis.

1.1 Automotive Engineering Design

The design process in automotive engineering is cyclic. Engineers virtually never start from scratch. New designs often evolve by making changes to previous ones. In an iteration, the effects of changes and new design ideas are evaluated and the design is refined based on the knowledge gained. Traditionally, new designs are evaluated by building physical prototypes and performing measurements on test bed systems.

There are several problems associated with development cycles involving prototype testing. Intense market competition requires that development costs are reduced and new designs reach production in a short time. Unfortunately, prototype production is expensive and time-consuming. Furthermore, there are certain physical attributes that cannot be directly measured on test beds, or only with insufficient accuracy, or only with limited spatial or temporal resolution. For example, direct and accurate measurements of gas temperature and flow velocity in the combustion chamber is not even remotely trivial.

[1] Greek philosopher and polymath, one of the most important founding figures in Western philosophy.

Alternatively, the data necessary to evaluate a design can also be acquired by computer simulation of physical phenomena in car and engine components. Virtually all aspects and components of automobiles can be simulated. Examples include mixture formation and combustion [20, 27], engine cooling [193], Diesel particulate filter regeneration [144], air conditioning in the passenger cabin and windscreen deicing [12], vibration and noise emission [97, 188, 225], and entire hybrid powertrains [70]. Testing new designs in simulation is more cost effective and allows shorter development cycles than making measurements on prototypes. Simulation can also compute attributes that cannot be measured in practice. Access to those attributes can facilitate more informed design decisions, which, in turn, can potentially improve product quality. This does not imply that testing prototypes on test beds can completely disappear from the design process [56, 198]. Computational simulation and test bed measurements complement each other at different phases of the design process. When possible, simulation models can be validated against test bed measurements [206].

Recent advances in computational resources have drastically reduced the time required for computing simulations. Accurate simulation models of complex systems can be computed rapidly. This allows the computation of many repeated simulations of the same model with different input parameters, within a reasonable time. Input parameters to the simulation include boundary conditions, for example, engine speed, external loads, and similar operating point parameters. Therefore, the system's behavior under different operating conditions can be examined. The input parameters can also reflect design choices and variants. For example, repeated simulations of a fuel injection system with different injection timing and fuel pressure parameters can be computed to evaluate the injection process in different design variants. Series of simulations produced by such parameter variations are called *multi-run*, or ensemble simulations [118, 203], and they are commonly performed in engineering [56, 95, 233], in climate research [181], and in other application domains. Accordingly, the resulting data sets, containing the parameters' value settings and results of all simulations, are called multi-run data sets. It is important to mention that while the mapping from simulation parameters to simulation results can be computed, the inverse computation is generally not possible [257]. It is usually impossible to explicitly determine the design parameters that produce a given set of simulation results.

The analysis of multi-run simulations offers interesting possibilities, because *parameter sensitivity analysis* [85, 91, 92] can be performed to study the relationships between the results and the parameters for different parameter value settings. Hamby [85] defines the following goals of parameter sensitivity analysis: identifying parameters that require additional research to reduce output uncertainty; identifying insignificant parameters that can be eliminated from the model; identifying parameters that contribute most to the output variability and are most highly correlated with the output; and investigating the consequences of changing a given input parameter.

When the model's sensitivity to the different parameters is known, then engineers can *optimize* the design to meet requirements. Design requirements are generally formulated in terms of expected simulation results. Therefore, the task is to find design parameters that produce desired results [257]. The results often depend on the parameters in a highly non-linear fashion, and small changes in the parameter values can cause profound changes in the results. Consequently, it is possible that small deviations from the optimal design parameter values produce

results that are very far from the designated target range. Unfortunately, such small deviations of design parameters are inevitable in manufacturing. In other words, design parameters can only be specified with a tolerance. Therefore, the engineer needs to define the design parameters such, that given the tolerances and the system's sensitivity to those parameters, the produced results always lie within the target range [257].

Computational data analysis methodologies, such as statistics, data mining, or machine learning are often used to analyze simulation data sets. Statistical methods [64, 176] and genetic algorithms [95, 233] are often used in optimization tasks. While computational methods are widely used and necessary, they are not always sufficient. They typically require that problems and interesting data features can be precisely defined from the beginning. The results of automated analysis of complex problems may be difficult to interpret. Exploring trends, patterns, relations, and dependencies in time-dependent data through statistical aggregates is not always intuitive. Certain prior knowledge of data patterns and properties of interest is required in order to compute useful statistical aggregates, and such knowledge is not always available. Without such knowledge, only using common statistical aggregates, important data features may remain completely hidden [11]. Data mining may fail to find interesting data features that appear natural to us [69]. Visualization and interactive visual analysis can support the analysis and knowledge generation from complex simulation data when computational methods prove insufficient.

1.2 Visualization and Interactive Visual Analysis

Card et al. [40] define visualization as "the use of computer-supported, interactive, visual representations of data to amplify cognition". Visualization utilizes the advanced human visual and cognitive system to support information drill-down in a guided human-computer dialogue [236]. There are three major goals of visualization [128]: (1) exploration, (2) analysis, and (3) presentation. This thesis mostly is concerned with the first two. Visualization for presentation usually demands different considerations that are not in the main scope of this work.

Exploration is generally the first stage in data investigation. Exploration involves searching for new, potentially useful information. The analyst tries to discover trends, patterns, clusters, outliers, and relationship in the data; and also check whether the data appear valid at all. Parts of the data set may need to be excluded, for example, because a simulation was non-converging. Visualizations need to offer a lot of flexibility and interaction to support exploration. During exploration, the analyst formulates hypotheses about the relationships and dependencies in the data.

The main goal of visualization for *analysis* (also called *confirmatory* visualization [228]) is to verify or reject those hypotheses; hence it is a more target-oriented drill-down for information. In the process, new questions and more refined hypotheses can be formulated, iteratively making use of the knowledge generated during analysis. Therefore, the analysis can alternate between exploratory and confirmatory tasks. In order to support analysis, the visualization software must allow the interactive formulation of visual queries that reflect the analyst's questions [236].

The goal of visualization for *presentation* is the effective visual communication and dissemination of the knowledge gained in analysis. The focus is not on knowledge discovery, but on the presentation of already known information. Therefore, visualization for presentation requires

different values compared to exploratory and confirmatory visualization. Interaction becomes less important, but the visual quality of the representations is essential. It is also often considered important that the visualization application is intuitive to use and does not require a long learning curve.

Interestingly, the three main goals of visualizations appear in history in exactly the opposite order as they usually follow in the analysis process. Presentation was the goal of most of the early visualizations in history; maps being obvious examples. Static confirmatory visualizations (charts) have been used in statistics for more than a century. Exploratory data analysis was promoted in Tukey's seminal book [253] in 1977. The origins of interactive, computer-aided visualization as a discipline are commonly dated to 1987 [169].

Interactive visual analysis (IVA) has evolved out of the fields of information and scientific visualization [127]. IVA is a multi-disciplinary approach to the analysis of complex data sets: it combines computational and interactive visual data analysis methods. Visual analysis facilitates the step-by-step data exploration and supports discovering unanticipated phenomena and relations in the data. In contrast to computational methods, it does not necessarily require that analysts explicitly formulate their questions [69], but promotes iterative knowledge discovery. Therefore, it can complement computational analysis methods. IVA is seen as a promising and valuable approach to the investigation of complex simulation data [88].

IVA systems can usually show different aspects of the data set in several distinct views [15]. Each individual view can be one of the commonly used representations in information visualization (histograms, scatter plots, parallel coordinates, etc.), or custom-built visualizations [131, 230]. If the data have an important spatial aspect, then spatial views, such as maps, volume rendering [6], flow visualizations [63], and other physical views [278] can be integrated.

The analyst can select subsets (often called *features of interest*) of the data, generally by visual means directly in one of the views. Data items of the selected subset are consistently highlighted in all views. This *linking-and-brushing* technique [21, 33] ensures that interesting parts of the data can be emphasized and the analyst can correlate the different perspectives of the features of interest. The visually emphasized subset represents the user's current focus. The rest of the data can be shown in a reduced form (in less detail, in less prominent color, etc.) to provide its context. *Focus+context* visualization [40, 87] helps the analyst to navigate in the data and to maintain his or her focus on the features of interest while also seeing its orientation with respect to the whole of the data. IVA, in its simplest form, involves brushing different parts of the data and comparing the different perspectives of the brushed subset.

Complex features cannot be defined by a single criterion on only one of the data attributes. Combined criteria on several data attributes are necessary to specify them [61]. Consequently, IVA systems generally provide means to combine brushes in several different views. Logical combinations of the individual brushes can be used to define features of interest over several data attributes [59]. Sometimes interesting data features cannot be directly expressed in terms of criteria directly on the attribute values. For example, exploring changes is often central to the analysis [10, 61], and that task can be supported by the computation of the first derivatives, or, in the discrete case, differences. Similarly, additional, synthetic data attributes can be computed by procedures from computational analysis such as principal component analysis or clustering. The power of the visual analysis is greatly increased by providing access to those derived, synthetic

attributes.

The integration of visual and computational methods is especially promising, because the best of both worlds can be combined [127]. Visual analysis tightly involves humans in the analysis. Humans are creative. We can select or invent suitable analysis strategies to derive knowledge. We can intuitively recognize interesting data features also in less well defined problems, which are often difficult to tackle with automated analysis methods. On the other hand, our cognitive capabilities have not increased significantly over time. Computers have become faster by several orders of magnitude in a few decades, but the same cannot be said about the human brain. The amount of information that the brain can process and analyze is limited. Unlike computers, humans are likely to make mistakes, especially when the same, monotonous tasks are performed.

Automated methods can process amounts of data that we cannot even effectively visualize. However, certain prior knowledge is usually necessary to design the automated analysis process—computers do not work by intuition. Automated analysis is also cheaper, compared to the labor costs of experienced analysts [129]. Hence it follows, that the strengths of the two approaches are complementary [25], and their combination can be a very powerful problem solving methodology. Tackling the same problems with a combination of the two approaches is expected to produce better results than the individual disciplines, in a more efficient way [121]. This realization led to the emergence of *visual analytics* [126, 128, 246, 247]. Visual analytics is a promising methodology to solve some of today's most pressing data analysis problems [129]. The combination of automated and visual analysis tools helps the analyst to synthesize information and derive insight from massive, dynamic, ambiguous, and often conflicting data; detect the expected and discover the unexpected; provide timely, defensible, and understandable assessments; and also to communicate assessment effectively for action [121, 126].

1.3 Contribution

This thesis is concerned with the interactive visual analysis of multi-run, multivariate, time-dependent simulation data sets in automotive engineering. The framework presented in this thesis is suitable for the analysis of data from a wide range of problem domains, including fuel injection, timing chain drive and elastic multibody simulations. The same principles have also been successfully used to analyze the evacuation of a building [74], a social network [137], and geospatial-temporal data [173].

The main contributions of this thesis advance certain aspects of the state of the art in the visual analysis of multi-run, multivariate, and time-dependent simulation data. We introduce a novel data model based on the concept of *families of function graphs* to represent simulation data effectively. We introduce *iterative composite brushing* to support step-by-step visual analysis. The *computation of derived data attributes and aggregates* is discussed as a method to find deeply hidden, implicit features in the data. We also address some of the specific requirements of visualization and visual analysis of *rigid and elastic multibody systems*. In the following, a summary of the main contributions is given.

Families of function graphs

Generally speaking, simulation computes values of physical quantities for each simulation time step. We use the term *function graph* to denote values of a single scalar quantity for different values of an independent variable. The independent variable is generally (but not necessarily) simulation time. It can also be frequency, or the even the index of a chain link in a chain motion simulation. A *family of function graphs* is a set of function graphs that represent the same quantity, but belonging to different simulation runs. The concept of families of function graphs is central to the analysis methods and techniques discussed in this thesis. We propose a data model which supports, in addition to scalar data attributes, function graphs as atomic data types. This data model significantly improves the analysis possibilities for time-dependent data, and it can also intuitively represent families of function graphs. Providing specific tools for the interactive visual analysis of function graphs (or curves, in general) is especially relevant, because many of the typical analysis tasks are related to the shapes and patterns in the curves. Those features are not easily defined in terms of explicit numeric values, therefore those tasks are difficult to tackle by computational methods.

We discuss several visualization techniques for families of function graphs. The *function graph view* is essentially a line chart that can simultaneously display all function graphs of a family. Overlaying many function graphs can cause visual clutter. We offer alpha-blending to improve the visual quality. Well-designed interaction features are important to support interactive analysis. We introduce the *line brush* as a tool to brush items in the function graph view intuitively and effectively.

When the independent variable is not continuous (frequency, for instance), then the continuous lines in the function graph view misleadingly suggest continuity. Furthermore, it is often difficult to choose the transparency factor in the function graph view that preserves outliers and makes details in crowded regions discernible at the same time. We propose the *segmented curve view* to overcome those limitations. We also introduce the *color lines view* which can effectively visualize clusters in families of function graphs, also with respect to other data attributes. All of these visualizations provide interaction and brushing features that support iterative visual analysis when embedded in a coordinated multiple views framework.

Multiple coordinated views and iterative composite brushing

IVA applications generally use multiple, coordinated views to display different aspects of the data. Most of the visualization and interaction techniques discussed in this thesis have been implemented in the flexible research prototype called ComVis [159]. A selection of well-known views from information visualization is offered, including histograms, scatter plots, and parallel coordinates; as well as views for families of function graphs. The combination of views and also the data attributes displayed in each view can be *freely configured*. Views can be temporarily maximized to help a more detailed examination. All views support *consistent focus+context visualization*: focus is shown in a bright color, while context is displayed in light gray. There is a linked table view of raw data values that supports quantitative assessments.

We offer *iterative composite brushing* to facilitate the flexible selection of data features. All views offer means of brushing data items. Several brushes can be defined in each view. Brushes

in the same or in different views can be combined using logical operators in an iterative manner. The new brush is used as the second argument of the Boolean operation, while the first argument is the current selection. This provides an intuitive way to narrow (AND, SUB operations) or broaden (OR operation) the selection. This simple approach has proven to be very effective in supporting interactive analysis. A color gradient can be applied to the brushed items. The color gradient is consistent over all view. This establishes visual links between the brushed items in different views.

The entire analysis status (data, view configuration, and brushes) can be saved and restored. Exchanging those session files facilitates collaboration among several analysts working on the same project. We have also found this feature very useful when collaborating on the publications related to this thesis.

Integrated computation of derived data attributes for families of function graphs

Not all interesting data features can be specified by combinations of brushes on the original data attributes. The definition of complex features may require that additional, synthetic data attributes are computed. The feature can be specified by brushing the synthetic attributes (compare to the work of Doleisch et al. [61]). We describe a framework to compute *derived attributes for families of function graphs*. This enables the analyst to specify features of interest that are not directly expressed in the data. The idea is inspired by curve sketching. In curve sketching, we aim to understand the shape of the curve. Attributes such as minimum, maximum, or zero-crossing are computed for a curve; as well as additional curves, such as the first derivative. We propose to compute similar derived data attributes, aggregates and curves for entire families of function graphs to support some typical analysis tasks. For example, finding function graphs of specific slopes can be supported by computing the first derivative and allowing the user to brush first derivative values. Engineers often want to find the function graph with the largest minimum or the smallest maximum in a family. Those features are usually occluded and not readily seen in a line chart. Computing the extrema of the function graphs generates scalar aggregates for each function graph. The scalar aggregates can be displayed in simple views (histograms, scatter plots) and the features can be brushed more easily.

The computation of derived attributes and aggregates is tightly integrated in the visual analysis system, so that it does not interrupt the analysis process. The derived attributes can be used in exactly the same way as the "original" data attributes. They can also be used as inputs to compute further derived attributes. With a sufficiently rich set of basis operations, including computation of first derivatives and integrals, curve smoothing, extrema, mean values, and percentiles, complex synthetic attributes can be derived interactively to support exploration of hidden, implicit data features. The aggregates that are found useful in an interactive analysis session can also be integrated into the design of computational analysis methods.

3D visualizations of rigid and elastic multibody systems

On the one hand, combinations of scatter plots, function graph views, and similar abstract views are useful in the analysis of relationships between different data variates. On the other hand, if the data set has a relevant spatial aspect, then additional views are necessary to provide the

spatial perspective. Different problem domains require different spatial visualizations. In this thesis, we present a framework for the visual analysis of multibody systems. We introduce *3D, glyph-based visualization for rigid multibody systems*, as well as glyphs to visualize scalar, vector, and rotational attributes of the motion. We provide several strategies of mapping data attributes to visual glyph properties to support different analysis tasks. Numeric values can be shown together with the glyphs on demand to support quantitative assessments.

We propose a similar, *glyph-enhanced visualization* for elastic multibody systems. We offer several techniques to counter occlusion—a serious concern in the visualization of elastic multibody systems. We also propose a method to improve the perception of torsional deformation, which is of special interest when the motion of rotating parts, such as crankshafts, is visualized.

1.4 Organization

The remaining parts of this thesis are organized as follows: Chapter 2 surveys the state of the art in fields related to the interactive visual analysis of multi-run, time-dependent, and multivariate simulation data. Section 3 describes a data model based on the concept of families of function graphs to represent simulation data sets effectively and efficiently. This chapter also introduces a coordinated multiple views framework with iterative composite brushing. Furthermore, generic analysis procedures are identified. In Chapter 4, we discuss three different levels of complexity we identified in visual analysis. Advanced brushing techniques and interactive computation of derived data attributes are discussed and compared as tools supporting complex analysis tasks. Chapter 5 introduces two novel visualization techniques for families of function graphs, as well as interaction features that support specific visual analysis tasks. Chapter 6 addresses the visual analysis of multibody systems, where data has a very relevant 3D spatial context.

A substantial part of this work has been done in cooperation with domain experts from the automotive industry. Chapter 7 documents the interactive visual analysis of simulation results of two engine subsystems, the fuel injection system and the timing chain drive. Both case studies result from our close collaboration with engineers and they demonstrate the applicability and usefulness of the methodology described in the previous chapters. Chapter 8 contains a summary of the work presented in this thesis. Chapter 9 provides some closing remarks. Acknowledgments and an extensive list of references conclude this thesis.

Chapter 2

Interactive Visual Analysis in Engineering, the State of the Art

"If you wish to make an apple pie from scratch, you must first invent the universe."

— Carl Sagan (1934–1996)[1]

In this chapter we survey the state of the art in the visual analysis of engineering simulation data sets. The structure of this chapter follows, in part, the classification by Kehrer and Hauser [118]. We discuss related work in visual analytics, in the visualization and visual analysis of time-dependent and multivariate data, as well as work on the comparative analysis of multiple simulations. We admit that the list of related work we review is by no means exhaustive. Several useful surveys are available in each of the fields mentioned here [3, 4, 36, 77, 126, 214, 283]. As a matter of course, they contain more in-depth reviews of the respective fields. We try to extract and present only the most relevant aspects with respect to the contribution of this thesis.

2.1 Interactive Visual Analysis

Interactive visual analysis is an approach to generating knowledge from large and complex data sets. It evolved from information visualization [127]; and it is an alternative to computational data analysis methodologies, such as statistics, machine learning, and data mining. Unfortunately, research has been evolving independently in visual and computational analysis. The two fields of science have remained relatively isolated, even though their goals are similar. Indeed, very promising synergies can be created by the integration of visual and computational methods [237], because the advantages and disadvantages of the two approaches are complementary [25]. This is also evidenced by the large volume of active, ongoing research [126, 127, 246, 247].

[1] American astronomer, astrophysicist, cosmologist, and science communicator in astronomy and natural sciences.

2.1.1 Visual Analytics

The aim of visual analytics, as defined by Thomas and Cook [246, 247] is to facilitate analytical reasoning supported by interactive visual interfaces. This very concise definition refers to analytical reasoning, a subfield in cognitive science, where there are many open questions [182, 270]. Therefore, Keim et al. [127] suggest a different, more specific definition:

> "Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex datasets."

Visual analytics evolved out of the fields of information visualization [128]. Visual data mining combines data mining techniques with visualization. There are several excellent surveys on information visualization and visual data mining by Keim [123], Keim et al. [130], and de Oliveira and Levkowitz [55]. According to which approach is more emphasized, Bertini and Lalanne [25] classify solutions into pure visualization, computationally enhanced visualization, visually enhanced mining, and integrated visualization and mining.

Compared to visual data mining, visual analytics is a more interdisciplinary science. It combines, among others, visualization, data mining, data management, machine learning, pattern extraction, statistics, cognitive and perceptual science, and human-computer interaction [246]. This rich combination of sophisticated methods from different disciplines enables the analysts to derive insight from complex, massive, and often conflicting data; detect the expected and discover the unexpected; find patterns and dependencies in the data; generate, reject or verify hypotheses; and also communicate the results of the analytical process [246]. Furthermore, tackling the same problems with a combination of visual and automated approaches can produce more accurate and more trustable results than the individual disciplines; and it can also be more efficient [121].

Visual approaches involve the human in the process, and that is not without disadvantages. It is human to make mistakes, especially when repeating the same task. It is cost-intensive to employ highly specialized experts [261]. Therefore, efficient automated analysis methods are often favored for well-defined problems, where the data properties are known and the analysis goals can be precisely specified [129]. Conversely, interactive visualization may be favored for vaguely defined problems [69] and also when the problem requires dynamic adaptation of the analysis solution, which is difficult to handle by an automated algorithm [129]. Findings from the visualizations can be used to steer the automated analysis [126], and, conversely, the knowledge gained from automated analysis can be used to generate more intelligent visualizations [155].

The visual analysis process generally follows the principles of Shneidermans visual information seeking mantra [236]: "*overview first, zoom and filter, then details-on-demand*". However, when the data set is large and/or very complex, then its direct visualization may be incapable of generating a useful overview, or it may not be possible at all. It becomes necessary to apply automated data reduction, aggregation, or abstraction before visualization. Two of the commonly used *data reduction techniques* are sampling [205] and filtering [123, 236]. *Data aggregation* methods include clustering [263], binning [43], and descriptive statistical moments [117].

Dimensionality reduction approaches reduce the dimensionality but attempt to preserve characteristics of high-dimensional data as good as possible. They include principal component analysis [111], multidimensional scaling [52], self-organizing maps [134], and feature extraction [200].

Keim et al. [127] define the visual analytics process as a transformation of data sets into insight, using interactive visualizations and automated analysis. The process begins with automated data transformations, including data cleansing, reduction, and aggregation. The resulting condensed data set preserves the important aspects of the data, and its size and complexity make it suitable for further analysis. This process is summarized in the visual analytics mantra [128]: "*Analyse First — Show the Important — Zoom, Filter and Analyse Further — Details on Demand*".

The condensed data set can be analyzed by visual means. The user can interact, select, zoom, and filter in the visualization; discover relationships, patterns, and gain insight directly from the visualization. The analyst can also generate hypotheses based on the visualization. He or she can evaluate them visually, or use computational tools to evaluate them, leading to new insights. Based on the insight gained, the analyst can request the computation of additional, synthetic data attributes [88], which can again be analyzed by visual or automated means. This leads to a useful feedback loop in the analysis process [126]. There are several visual analysis systems that integrate the computation of statistics and derived data attributes [58, 93, 120, 136, 194, 277].

The purpose of the analysis process is gaining insight [40]. It follows, that the success of analysis tools can be estimated by measuring the insight gained. However, the definition of insight often remains fairly informal, making success difficult to measure [183]. Yi at el. [289] describe four categories of insight gaining processes. Understanding what insight is about can make us able to design systems that promote insights, and also to evaluate analysis systems in an insight-based manner. Studies conducted within the visualization and visual analytics community are often limited to a relatively short period of observation time and therefore fail to capture the long-term analysis process. Saraiya et al. [223] have presented a longitudinal study of analysis of bioinformatics data. The paper documents the entire process (over one month) from the raw data set to the insights generated. Such studies (see also González and Kobsa [81]) can enhance our understanding of visual analytics and provide guidelines for future development.

Keim et al. [128], as well as González and Kobsa [81] emphasize that visual analysis tools should not stand alone, but should integrate seamlessly into the applications of diverse domains, and allow interaction with other already existing systems. Sedlmair et al. [231] report their experiences in integrating novel visual analysis tools into an industrial environment. Chen [47] discusses visual analytics from an information-theoretic perspective.

2.1.2 Coordinated Multiple Views

There is usually no single visual representation that can display all of relevant aspects of complex data sets. Interactive visual analysis systems often combine different views on the same data in such a way that a user can correlate the different views. The survey by Roberts [214] provides an overview on the state of the art in *coordinated multiple views* (CMV). There are a lot of well-known visualization systems based on the CMV approach, including GGobi [243], Improvise [274], Mondrian [245], SimVis [58], Snap-Together Visualization [184], Visplore [197],

WEAVE [83], and XmdvTool [220]. Baldonado et al. [15] suggest that multiple views should be used when the data attributes are diverse, or when different views can highlight correlations or disparities. Smaller, manageable views can also help decompose the data into manageable chunks. They also point out that multiple views demand increased cognitive attention from the user and introduce additional system complexity.

Individual views in a CMV system can display different dimensions, subsets, or aggregates of the data, thus the visualization can follow a "divide-and-conquer" approach. General purpose CMV systems usually offer a selection of *attribute views* [118] well-known from information visualization, including bar charts, scatter plots [251, 195], and parallel coordinates [101, 106, 108, 185]. Time-dependent data can be displayed in line charts [96, 178]. Systems targeted at specific problem domains can incorporate specialized views [131, 230]. Systems for the analysis of data with a relevant spatial context (e.g., flow simulation or CT scans) can integrate 3D views [58, 83, 116].

CMV systems can be categorized based on the number of views they manage. On the one hand, *dual view systems* [51] combine only two views of the data set. For example, one view can provide overview while the other shows details; or one view can be used to control the other. On the other hand, general *multi-view* environments allow any number of views to be created. Most commonly, views are created using standard menus or buttons [2, 184, 245]. One can also attempt to find expressive visualizations in a (semi-)automatic manner [156]. For example, Tableau [157] and Visage [218] can create a set of views based on data characteristics and user preferences. As the number of linked views and the amount of coordination increases, it may become necessary to visualize how the views are linked [184, 275].

Efficient interaction with the visualization is crucial in the analysis process [213, 288]. Relationships between data attributes can be detected visually if interesting parts of the data set can be selected and the related items are consistently highlighted in linked views [33]. The selection is typically defined directly in the views by brushing [21]. Brushing and linking effectively creates a *focus+context* [50, 87, 178] visualization where the selection is in focus and the rest of the data set provides its context. Complex queries can be expressed by logical combinations of several brushes [158]. Brushes can be combined via a feature definition language [59], or in conjunctive visual forms [276].

The selection in most systems is binary: a data item is either selected by the brush or not. This is not always beneficial. Flow simulation data, for instance, often exhibits a rather smooth distribution of attribute values in space. This smooth nature is reflected in *smooth brushing* [60], which results in a continuous degree-of-interest (DOI) function. The DOI can also be interpreted as the degree of being in focus, analogous to generalized fisheye views [79]. The continuous DOI function can be used for opacity modulation in the linked views; thereby smooth focus+context visualization is achieved. Muigg et al. [178] propose a four-level focus+context visualization, consisting of three different kinds of focus and the context.

2.2 Time-Dependent Data

Computer simulations or repeated measurements generate time-dependent data in many different disciplines, including engineering [6, 62], medicine [160, 187], climate research [119],

meteorology, and economics. In certain cases, time can be treated as one of the quantitative dimensions and displayed as such in parallel coordinates or in scatter plots, for example. This simple approach may even outperform specialized time-dependent data visualizations for some basic analysis tasks [3]. Time, however, is an outstanding dimension with particular meaning and properties that generally need to be reflected in the visualization in order to support analysis. Due to the special role of time, several books [5, 10] and surveys have been published on the visualization [3, 179, 240] and visual analysis [4] of time-dependent data.

Aigner et al. [3] propose systematic categorizations of time-dependent data. The first aspect in their classification reflects the characteristics of the time axis. The primitives on the time axis are either points or intervals. The ordering of the primitives can be linear, cyclic, or branching. Frank [73] also suggests the concept of multiple perspectives to describe events on different time axes. Aigner et al. [3] also classify the data with respect to the frame of reference (spatial vs. abstract) and the number of variables (univariate vs. multivariate). They also differentiate between the visualization of data per se and abstractions thereof. Increasingly higher levels of abstractions are classified as aggregates [264], features [208], and events [66].

In the following we review visualization techniques and visual analysis methods for time-dependent data.

2.2.1 Visualization Techniques for Time-Dependent Data

The visualization techniques for time-dependent data can be classified into two distinct groups based on whether or not the visual representation itself is time-dependent [3, 179]. Dynamic (time-dependent) visualizations depict time directly by automatically changing the visual representation over physical time, essentially producing an animated view. On the contrary, static visualization techniques do not automatically change over time. Whether user interaction can change the visualization is immaterial to this categorization. Dynamic visualizations can often provide a general overview of the data and support qualitative assessments. For instance, many flow visualization techniques display time-varying flow via animation [62, 208, 260]. Unfortunately, humans find it difficult to perceive and conceptualize animations accurately [254], especially when long time series are animated. In visual analysis, static representations are often preferred when quantitative assessments need to be made [179].

Several visualization techniques originally not designed for time-dependent data have been enhanced to depict time. *Small multiples* [252] use multiples of a chart, each capturing an incremental moment in time. Each image must be interpreted separately, and side-by-side comparisons must be made to detect differences. This is only feasible for short time series. *Time-Histograms* [142, 290] display histograms of scalar data at each time step either in 3D as a row of cuboids, or in 2D as an image. Wegenkittl et al. [280] proposed parallel coordinates extruded in 3D for the visualization of high dimensional time-dependent data. Blaas et al. [28] and Johansson et al. [107] propose using transfer functions in parallel coordinates to visualize time-dependent data.

A lot of special visual metaphors have been proposed for time-dependent data. In the scope of this thesis, the visualization of multivariate time-dependent data is particularly relevant. Javed et al. [103] surveyed several line graph techniques involving multiple time series and compared user performance for comparison, slope, and discrimination tasks. The *Line Graph Ex-*

plorer [132] provides a compact overview of a set of line graphs. The Y dimension is encoded using color, thus a line graph is represented by a thin row of pixels. Rows representing individual line graphs are packed tightly over one another to display an overview of the entire set. Using color instead of vertical height (as in traditional line graphs) reduces the level of perceivable detail. The authors propose focus+context visualization to compensate for that: selected lines can be expanded and shown as traditional line graphs. A similar visualization by Peng [190] does not use a continuous color gradient, but represents low, middle, and high values in time series in three discrete colors. The two-tone color mapping [221] uses two neighboring colors of a color map in several rows of pixels instead of only one. This communicates the values with more accuracy and also makes the slope of the line graph more visible. Daae Lampe and Hauser [53] propose a technique based on kernel density estimation for rendering smooth curves also with a frequency higher than the pixel width. Transitions between high frequency areas (context) and single line curves (recent values) are also smooth.

Tominski et al. [249] have proposed two radial layouts of axes for the visualization of multivariate time-dependent data: the *TimeWheel* and the *MultiComb*. *Lexis pencils* [72] display several time-dependent variables on the faces of a pencil. Pencils can be positioned in space to indicate the spatial context of the data. Tominski et al. [250] have used 3D icons on a map to visualize linear or cyclic patterns in time series in a spatial context. Kapler and Wright [115] propose a 3D visualization of time-dependent data where the X-Y plane provides geospatial information and time is represented along the Z axis. The ground plane marks the instant of focus. Past events are shown under the ground plane, future events over the plane. Andrienko and Andrienko [9] have proposed an aggregation based approach for the visualization of proportions in spatio-temporal data.

The *ThemeRiver* [90, 273] visualizes changes in topics in large document collections. The frequency of certain topics is depicted by colored bands that narrow or widen to indicated changes in the frequency. Byron and Wattenberg [39] survey similar stacked graph techniques, also considering aesthetics and legibility. Spiral layouts have been proposed [42, 279] as means of highlighting periodic and cyclic patterns in time series.

2.2.2 Visual Analysis of Time-Dependent Data

Andrienko and Andrienko [10] categorize common analysis tasks associated with spatio-temporal data, such as relation and pattern seeking, lookup and comparison. Aigner et al. [4] survey visual analysis methods for time-oriented data.

SimVis [62] has been adapted for the analysis of time-dependent simulation data. Each view can either show data of one time step or accumulate the data of many successive time steps. A two-level focus and context visualization is implemented. The first level is a traditional focus and context view for data in the currently active time steps. The second level of context displays the data of all time steps. Akiba and Ma [6] propose a system where time-dependent flow features can be explored using a combination of time histograms, parallel coordinates, and volume rendering. This approach effectively partitions the three factors contributing to the complexity of the data into three views: (1) time histograms display the time-dependent nature of the data, (2) parallel coordinates display multivariate data, (3) the volume rendering provides spatial details. Fang et al. [67] represent time-varying 3D data as an array of voxels where each

voxel contains a time-dependent value, a time-activity curve (TAC). The volume visualization uses transfer functions based on similarity between TACs. The authors propose three similarity measures: similarity to a template TAC (represented in a 1D histogram), similarity and Euclidean distance to the template TAC (represented in a 2D histogram), and similarities between all pairs of TACs (represented in a 2D scatter plot via multidimensional scaling). The user can explore the time-dependent volume by brushing the respective similarity measures.

TimeSearcher [96] is a well-known tool for the interactive exploration of time-dependent data. Combinations of timebox widgets can be used to brush both the time axis and the attribute axis. Changes in the time series can be found by angular queries. When large sets of function graphs are analyzed, it is necessary to compare them against a certain pattern. Similarity brushing [34, 35] enables the user to brush all time series similar to a selected one. In QuerySketch [272], the user can draw a sketch of a time series profile and similar time series are retrieved, with similarity defined by Euclidean distance. Muigg et al. [178] allow the user to sketch a polyline approximation of the desired shape. Frequency binmaps [185] are used to aggregate function graphs and maintain performance with larger data sets. Visual clutter is reduced by drawing pixels through which more function graphs pass in higher luminance. LiveRAC [171] uses a reorderable matrix of charts, with semantic zooming adapting each chart's visual representation to the available space. Side-by-side visual comparison of arbitrary groupings of devices and parameters at multiple levels of detail is possible.

Aigner et al. [4] point out that the analysis of larger volumes of time-oriented data can be facilitated by combining visual and analytical methods, such as aggregation, temporal data abstraction, principal component analysis, and clustering (compare to the visual analytics mantra [128]). López et al. [264] survey aggregation approaches for spatiotemporal data.

The *Calendar View* [263] groups time series data into clusters, effectively displaying trends and repetitive patterns on different time scales in univariate data. VizTree [152] transforms the time series into a tree in which the frequency and other properties of patterns are mapped to color and other visual properties. It provides interactive solutions to pattern discovery problems, including the discovery of frequently occurring patterns (motif discovery), surprising patterns (anomaly detection), and query by content. Hao et al. [86] describe a system to explore potentially overlapping motifs in large multivariate time series. The visualization of the time series can be distorted to emphasize the motifs discovered in automated motif mining, or their context. Bak et al. [14] analyze animals' movement using hierarchical clustering in the time domain and growth ring maps to manage overlapping in space. Temporal summaries, proposed by Wang et al. [265], dynamically aggregate events in multiple granularities (year, month, week, etc.) for the purpose of spotting trends over time and comparing several groups of records.

Zhang et al. [292] introduce the first Fourier harmonic projection to transform the multivariate time series data into a two dimensional scatter plot. The spatial relationship of the points reflects the structure of the original data set, and relationships among clusters become two dimensional. Woodring and Shen [285] use wavelet transformation to transform time-dependent data to a multiresolution temporal representation, which is then clustered to derive groups of similar trends. The user can make adjustments to the data in the cluster through brushing and linking. Ward and Guo [269] map small sections of the series into a high-dimensional shape space, followed by a dimensionality reduction process to allow projection into screen space.

Glyphs are used to convey the shapes. Interactive remapping, filtering, selection, and linking to other visualizations assist the user in revealing features, such as cycles of varying duration and values, anomalies, and trends at multiple scales.

Oeltze et al. [187] integrate correlation analysis and principal component analysis to improve the understanding of the inter-parameter relations in perfusion data. Voxel-wise temporal perfusion parameters and principal components can be jointly analyzed using brushing and linking to specify features. The authors demonstrate their approach in the diagnosis of ischemic stroke, breast cancer, and coronary heart disease. Kehrer et al. [119] derive temporal characteristics such as linear trends or signal-to-noise ratio and enable the user to brush them to steer the generation of hypotheses.

2.3 Multivariate Data

Simulation data sets contain the values of the simulation control parameters that represent the boundary conditions and choices of design parameters. They are independent variables from the perspective of the simulation process. The results of the simulation depend on the values of the independent variables. Typically, many different data attributes are computed simultaneously. Therefore, simulation data sets are of high dimensionality. The visualization and analysis of high dimensional data sets has a long history. Accordingly, there is a vast body of related literature [77, 284]. Wong and Bergeron [284] suggest using the term *multidimensional* to refer to the dimensionality of the independent variables. The term *multivariate* refers to the dimensionality of the dependent variables. Bürger and Hauser [36] classify multivariate data visualization techniques by data dimensionality and based on the stages of the visualization pipeline at which they take effect.

2.3.1 Multivariate Data Visualization

In this section we discuss visualization techniques specifically designed to display multivariate data in a single view. An alternative to displaying all variates in one view is showing subsets of the variates (projections of the data set) in coordinated multiple views. Indeed, when data is of very high dimensionality, then that is often the only feasible approach. Keim [123] classifies visualization techniques of high dimensional data into the following groups: geometric projections, iconic techniques, pixel-based techniques, and hierarchical methods.

Geometric projections attempt to provide informative projections of multivariate data sets. Geometric projections include many of the well-known, traditional views in information visualization. Scatter plots [251] are one of the oldest and most commonly used projections. Correlations between more than two dimensions can be explored by arranging scatter plots in a matrix [49], using a Hyperbox [8], or the HyperSlice [262]. The Prosection Matrix [80, 257] projects data points in the vicinity of the 2D slices to scatter plots. There are several ways to encode more than two dimensions in scatter plots by using symbols or glyphs instead of points, or by modulating the points' size or color [195]. Scatter plots can be extended into 3D [143], but the issues related to occlusion, comprehension and interaction difficulties need to be addressed [195].

The conventional layout of axes representing dimensions is orthogonal, which allows a maximum of only three axes. There are many techniques that suggest a different, potentially non-orthogonal layout of more than three axes. Parallel coordinates [100, 101, 291] arrange axes side-by-side vertically. The ordering of the axes has a major impact on the expressiveness of the visualization [106, 286]. Different orderings highlight different aspects and correlations in the data and produce a varying amount of visual clutter [191]. The visual clutter due to overdraw when plotting many data items can be reduced by binning [185], clustering [76], and transfer functions on high-precision textures [108]. Zhou et al. [293] use curved lines in parallel coordinates. The lines form visual bundles that are perceived as clusters. In star coordinates [99, 112], axes radiate out from the center of a circle and extend to the perimeter. The visualization is actually star glyphs superimposed over each other. In Radviz [24, 98], dimensions are represented by anchors arranged around a circle. Data points are connected with springs to each of the anchor points.

Pixel-oriented techniques [122] map data items to colored pixels. Each dimension is usually presented in a separate subwindow, thus each pixel represents one attribute value. The subwindows are usually rectangular and they are arranged in a matrix, but circle segments arranged in a disk have also been proposed [122]. The problem of finding a good layout of the subwindows representing dimensions is similar to that of ordering the axes in the parallel coordinates. Pixel bar charts [124, 125] visualize additional data attributes within bars of a bar chart. The color scale that maps attribute values to color should be intuitive for the application domain [122], and, at the same time, should be perceptually uniform [149, 215]. The gray scale, for instance, has the nice property that it increases monotonically in luminance, but the number of just noticeable differences in gray is only 60–90 [149]. The often used rainbow color map is plagued with a number of perceptual problems [30, 215], such as lack of perceptual ordering, artificial color gradients that do not represent contours in the data, and uncontrolled luminance variation. The heated object color scale [151] and the linearized optimal color scale [150] are perceptually uniform and perform better than the gray scale. Keim [122] proposed creating color maps by linear interpolation in a special hue-saturation-intensity color model.

The arrangement of pixels within a subwindow is important, because only good arrangements allow the discovery of clusters and correlations. If there is some natural ordering in the data, then that can be used as a guideline for the arrangement of pixels. If some two dimensional ordering exists in the data, then the mapping can be trivial. Space-filling curves [122] have been proposed as a method of mapping the one dimensional ordering in the data to a two dimensional array of pixels. The recursive pattern technique [122] recursively organizes pixels in groups. For example, with time series data, the grouping can follow the natural grouping of days, weeks, months, and years; in order to highlight patterns of different time scales. Alternatively, data items can be ordered based on their distances from the user's query. This is especially relevant when only data relevant in the context of a specific query are visualized, or when there is no natural ordering in the data, but distance between two data items can be defined. Space-filling curves and recursive pattern techniques do not intuitively convey the notion of distance. Spiral layouts and combinations of spirals and space-filling curves [122] preserve both ordering and clusters. Items satisfying the query are positioned in the middle. Items approximately matching the query are shown further away from the center, according to their distance to the query.

Iconic, or glyph-based techniques [268] map multivariate data items to icons. Examples of well-known iconic visualizations include Chernoff faces [48], stick figures [192], star glyphs [44] (comparable to star coordinates [99]), color icons [148], Lexis pencils [72], and specialized glyphs proposed for flow visualization [54, 199] and other custom-built glyphs [209].

In iconic visualizations, data attributes are mapped to visual properties (orientation, size, shape, curvature, color, etc.) of the icons. Ward [268] classifies mapping into three categories. One-to-one mappings map each data attribute to one distinct visual property. One-to-many mappings create redundancy, which can make the visualization more accurate or easier to interpret. Many-to-one mappings display several data attributes using the same visual property, separated in space, orientation or other transformation. Certain visual properties are processed preattentively by the human visual system [270], and information encoded into those properties "pops out". It sounds logical that the most important data attributes should be mapped to visual properties that are most accurately assessed, for example, length and angle [156]. Nowell and her colleagues [186] suggest that the best choice of graphical encoding depends on the nature of the task. Naturally, the user's domain knowledge can also be considered when searching for intuitive mappings. Ward [268] suggests four strategies (correlation-driver, symmetry-driven, data-driven, and user-driven) to define the order in which dimensions are mapped to graphical properties. Some properties and relationships between properties are easier and more accurately perceived than others, hence, depending on the mapping, the assessment of relationships between dimensions can suffer from bias.

The placement of glyphs can encode significant information. Ward [267] suggests several guiding principles to classify glyph placement strategies. Data-driven strategies position glyphs based on original or derived data values. Structure-driven placement methods are based on an explicit or implicit order or other relationships between data items. If glyphs are allowed to overlap, then this can have an impact on the quality of the visualization. On the other hand, if overlapping is not allowed, then the size of glyphs or the number of data items is limited. The placement can be space-filling, or use the empty space between glyphs to express distance between data items. A further consideration is whether the computed initial placement can be adjusted interactively.

Ropinski et al. [216, 217] propose a glyph taxonomy based on the theory of perception, instead of technical properties. They distinguish preattentive stimuli (shape, color, placement) that provide a first impression, and attentive stimuli (composite glyph shapes, glyph legends, interaction), that provide more quantitative detail. They suggest that the attributes should be mapped to glyphs such, that the users' attention is focused to the important ones, and glyphs should support quantitative analysis in the attentive phase. The authors also suggest glyph usage guidelines. For instance, 3D glyph shapes should be unambiguously perceivable independent of the viewing direction. The superquadric glyphs proposed by Jankun-Kelly [102] and Kindlmann [133] satisfy this criterion. Kehrer et al. [120] place glyphs as billboards in 3D space to avoid ambiguity depending on the viewing direction.

Hierarchical methods subdivide the data space and present subspaces in a hierarchical fashion. The resulting visualization does not treat dimensions equally. Different hierarchies produce different views of the data, featuring different dimensions as more important. Users may need training to be able to interpret the visualization [55]. An example of a one dimensional hierarchy

is the hierarchical axis [172]. Dimensional stacking [147, 266] is a well-known example in two dimensions and works well for discrete categorical data. In "worlds within worlds" [68], each location in a 3D space can contain another nested 3D space that the user can explore in a hierarchical fashion to analyze an n-dimensional function. Treemaps [235] and mosaic plots [75] hierarchically partition the window into regions, depending on attribute values.

2.3.2 Visual Analysis of Multivariate Data

One of the principal challenges in the visual analysis of multivariate data is *finding relationships* between different data dimensions. Multivariate data is often analyzed using coordinated multiple views (see Section 2.1.2). Brushing and linking between the views establish visual links between dimensions, thus the user can discover relationships. Some systems, such as SimVis [58] and WEAVE [83], incorporate 3D visualizations. This enables the user to investigate spatial correlations and localize features of interest in space. The Attribute Explorer [256, 242] uses linked histograms to simultaneously represent the interactions between attributes and allow the user to narrow the focus by defining limits on certain attributes.

Allowing the user to *reorder* [23] columns and rows in a matrix layout sounds trivial, but it facilitates surprisingly powerful data exploration [239]. Parallel coordinates are often used to display high dimensional data. The perceived correlations between dimensions depend on the order of axes [106]. Therefore, interactive reordering of axes [28] can help the user in discovering them. Hauser et al. [89] introduce angular brushing in parallel coordinates. Correlations between neighboring axes can be discovered by selecting the line segments of similar slopes between axes. Novotný and Hauser [185] describe an enhancement of parallel coordinates that preserves both outliers and main trends. Each line segment between adjoining axes contributes to one bin of a 2D binmap. Clusters and outliers are detected in the binmaps, and this information is used to render trends and outliers. Johansson et al. [108] utilize clustering and apply different transfer functions to a high-precision texture to emphasize outliers, or facilitate the simultaneous analysis of many clusters.

Dimensions can be ordered in a computer-assisted manner to (semi-)automatically generate useful visualizations. Peng et al. [191] propose measures of visual clutter for four popular visualization techniques and aim to improve the expressiveness of the visualization by automatically reordering the dimensions to minimize clutter. The Rank-by-Feature Framework [234] presents projections of multidimensional data sets in histograms and scatter plots. The user can order the visualizations by the statistical moments of the displayed dimensions. This framework enables the user to find interesting projections. Tatu et al. [244] propose measures to rank scatter plots and parallel coordinates to generate potentially useful visualizations.

Dimensionality reduction is often applied to multivariate data during analysis. The grand tour technique [13, 32] projects multidimensional data onto two dimensional planes. The projection planes can be moved in the high dimensional data space to generate an animated view. Automated dimensionality reduction, such as principal component analysis and multidimensional scaling may generate lower dimensional spaces that have little intuitive meaning. Jeong et al. [104] developed a systems that visualizes the results of principal component analysis and offers interaction features that assist the user in better understanding and utilizing PCA. The visual hierarchical dimension reduction proposed by Yang et al. [287] enables the user to control

the dimension reduction process interactively, in order to generate lower dimensional spaces that are meaningful to the analyst. In their later work [286], the authors also filter out dimensions that are similar to others or uninteresting for the user's visualization task. The filtering automatically generates an initial result that the user can modify interactively.

The specification of flow features (e.g., vortices) typically requires criteria spanning several dimensions. Henze [93] uses "portraits" in linked derived spaces. The portraits are essentially enhanced scatter plots that display pairs of flow attributes. A rich selection of brush types can be used to select regions of interest in each portrait. In *SimVis* [59], compositions of smooth brushes in scatter plots and histograms can be used to specify flow features. The specifications are expressed in an XML-based feature definition language and are persistent across analysis sessions. Feature extraction is a method of reducing high dimensional data. Bürger et al. [37, 38] integrate multiple local vortex detectors into an interactive flow feature detection system. The user can aggregate the information from the individual detectors to combine their advantages and increase the credibility of the feature extraction. In their more recent work, Fuchs et al. [78] combine machine learning and visual analysis. The user creates an initial hypothesis via brushing and linking, and a heuristic search algorithm finds alternative or related hypotheses. The hypotheses are represented as a set of fuzzy selections in the visualization. The generated hypotheses are automatically added to the visualization and are available for further refinement.

2.4 Multi-run Data

Multi-run simulation data is produced when the simulation is repeated with perturbed parameter settings to represent different design choices or boundary conditions. The idea is rooted in Monte-Carlo simulations [26, 177], dating back to the 1940s. With recent increase in computational power, accurate multi-run simulations of complex systems can be computed in reasonable time. Multi-run simulations are commonly used in many different application domains, for example, in automotive industry [163, 194], climate research [181], biology [227], and epidemiology [1].

The analysis of multi-run simulations is of interest for engineers because they can study how parameters influence the behavior of the simulated system and choose optimal parameter combinations. In *parameter sensitivity analysis* [85, 91, 92], the significance of parameters is determined and their correlations with the output is studied. When the model's sensitivity to its parameters is understood, engineers can *optimize* the design by appropriately choosing parameters.

In this section, we discuss possible approaches to the visualization and visual analysis of multi-run simulation data. An alternative to multi-run visualization is the very actively researched field of *uncertainty visualization*. There are several surveys and state of the art reports on uncertainty visualization by Brodlie et al. [31], Griethe and Schumann [84], Johnson and Sanderson [110], and Pang et al. [189].

2.4.1 Visualization of Multi-run Data

Visualization of multi-run data sets is especially challenging, because they are typically high-dimensional, contain several different time-dependent result attributes (independent variables), and are also multivariate in the sense that multiple data values are given for each position in space and time, which complicates direct spatio-temporal visualization.

Data pertaining to different runs can be directly displayed in small multiples [203] or in a coordinated multiple views framework. Abstract information visualization views as well as detailed spatio-temporal views can be incorporated. However, this approach is limited to displaying only a few runs at a time. If the data set has no relevant spatial aspect, then the simulation control parameters can be considered independent variables in the data set and multi-run data can be compactly visualized using techniques for high-dimensional data [166, 194]. Alternatively, one can visualize statistical measures of the individual distributions of the time-dependent attributes from multiple simulation runs [181, 202].

The standard representation of distributions in statistics is the *box plot* [170, 253]. Box plots display the minimum and maximum data values, the median, and the lower and upper quartiles. Unlike histograms and probability density functions, box plots do not require assumptions of the statistical distribution. Potter [201] surveys several modifications of box plots that provide additional information, including the sample size, density information and further statistical properties, such as skewness and kurtosis. Hintze and Nelson [94] propose a combination of box plots and density shapes called violin plots. Potter et al. [202] extend violin plots by a color mapped histogram and glyphs representing additional descriptive statistics such as mean, standard deviation, and higher order moments.

The box plot can also be used for higher dimensional data. The main problem is how to represent the summary values using simple visual metaphors with meaningful spatial positions. Proposed extensions of box plots for bivariate distribution include the rangefinder box plot, two-dimensional box plot, bagplot, relplot and quelplot [201]. Potter et al. [202] present an extension of their summary plot to 2D distributions.

Box plots cannot be placed in a dense spatial context. Kao et al. [113, 114] display summary statistics of 2D distributions on surfaces using color mapping and line glyphs. This is sufficient when the distribution can be adequately characterized by its statistical parameters. When that is not possible, the authors create a 3D volume from the 2D distribution by adding the data range as a third dimension. The voxel values represent the probability density function. Standard volume visualization techniques can be applied to this density estimate volume. Moreover, shape descriptors (number, location, height, and width of peaks) are computed from the density estimate volumes and can also be visualized by colored slices and line glyphs.

Kehrer et al. [120] represent statistical properties computed from the multi-run data using billboard glyphs placed in a 3D context. Sanyal et al. [222] combine glyph-based uncertainty visualization with the spaghetti plots to display ensemble meteorological simulations. Spaghetti plots [57] combine contours of an attribute at a selected time step over all simulation runs. They are commonly used in meteorology.

Chan et al. [45] represent sensitivity parameters as tangent lines in 2D scatter plots. This helps analysts discover local and global trends in a 2D projection. Changes in one variable can be correlated to changes in another one. The sensitivity information can be understood

as velocity, and the resulting visualization resembles a flow field. The artificial flow field can be explored also based on flow-field analysis. For example, data points can be selected and clustered by streamlines. This groups points with similar local trends in a non-linear manner.

2.4.2 Visual Analysis of Multi-run Data

In some cases, multi-run data can be analyzed using methods that were originally designed for high-dimensional data. The HyperSlice [262] displays a multi-dimensional function as a matrix of orthogonal two-dimensional slices around a focus point in the multi-dimensional space. The Influence Explorer [257] allows the exploration of data computed from a model using given sets of parameter values as input values. The user can select a set of points in either the parameter space or the result space and see how that set corresponds to points in other dimensions in both spaces. The Prosection Matrix [80, 257] allows the user to define a slice of adjustable thickness (representing tolerance) and projects data points in the slice to scatter plots. Matković et al.[166] describe a system based on coordinate multiple attribute views and interactive brushing that can be used for the analysis of multi-run simulations. The work presented in Chapters 3, 4, and 5 of this thesis extends this framework. Nocke et al. [181] analyze statistical aggregates of multi-run climate simulations in multiple linked attribute views.

Multi-run data can be analyzed based on statistical moments. The system proposed by Potter et al. [203] for spatio-temporal multi-run data links overview and statistical displays. Kehrer et al. [117] integrate the computation of higher-order statistical moments like skewness and kurtosis as well as robust estimates in the visual analysis process and enable the analyst to brush particular statistics. In more recent work by Kehrer et al. [120], multi-run data and aggregated properties are related via an interface, so that selection information between the data parts can be communicated. The statistical properties are represented using glyphs. The glyphs can be explored using focus+context visualization and brushing of statistical aggregates.

The results of time-consuming multi-run simulations can be approximated by simpler surrogate models. The HyperMoVal [194] enables the user to validate such models visually. The system shows multiple 2D and 3D projections of the n-dimensional function space around a focal point. The predictions can be compared to known simulation results. The validity of the model at different points can be analyzed, thus regions of bad fit can be found. Unger et al. [258] discuss a framework for the validation of geoscientific simulation ensembles.

Berger et al. [22] propose a tool, based on HyperMoVal, for the continuous exploration of a sampled space of simulation input parameters and results. The neighborhood of a selected point in the input parameter space can be projected to the result space using nearest-neighbor or model-based predictors. The inverse mapping, from results to parameters, is generally not possible. The authors visualize the neighborhood around the selected point where changes in predicted output produced by the variations of the input parameter remain below a threshold. The uncertainty of the predictions is visualized by box plots.

Matković et al. [164] visualize families of data surfaces, i.e., data sets with respect to pairs of independent dimensions. Different levels of abstraction (scalar aggregates, projections with respect to one variable) can be analyzed using multiple views and brushing. Direct visualization of surfaces is used only when the user drills down to one (or a few) interesting surface(s), in order to avoid problems related to occlusion. Piringer et al. [196] analyze and compare surfaces

using multiple levels of detail and tight integration of domain-oriented and member-oriented overviews. Alabi et al. [7] compare slices of surfaces from an ensemble in a single image view.

Combining data visualization with the visualization of the process that generates data can significantly improve the understanding of the data, as well as the underlying process [259]. Unger and Schumann [259] describe a framework that enables the users to explore the simulation process at three different levels. Visualization on the experiment level relates the multi-run simulation data to the model structure and the experiment. The visualization on the model level coordinates multiple instances of this view to facilitate the comparison of experiments. On the level of multi-run simulation data, the view provides an overview, and details are shown in time series views. Matković et al. [161] integrate the simulation model view in their coordinated multiple views analysis framework. The simulation model view provides a visual outline of the simulation process and the corresponding simulation model during analysis. In order to use display space efficiently, data can be visualized at three different levels of detail (histograms, scatter plots, and curve view).

Interactive visual steering of the simulation represents a tighter integration between visualization and simulation (also compare to Johnson [109]). Matković et al. [163] propose an approach where the user can initiate new simulations with input parameters defined interactively via brushing. This approach allows the rapid exploration of the parameters space with increased resolution in interactively identified regions of interest; and also provides a deeper understanding of the system's behavior. In their more recent work [162], the authors incorporate optimization based on a regression model and interactively defined objectives. The optimization automatically suggests simulation parameter values for the next iteration.

Ribičić et al. [210] present a tool for the real-time analysis of interactively steered simulations. Overview is provided by data aggregates embedded into the simulation rendering. Details can be analyzed in linked range views. A selection mechanism connects the two approaches. Points of interest are selected by clicking on aggregates, supplying data for the detail views. This allows the user to maintain an overview of the ensemble and perform analysis even as new data is supplied through simulation steering. Alternative simulation scenarios are represented by World Lines [271]. New branches are created when the user modifies simulation parameters. More recently, Ribičić et al. [211] propose a novel way to steer simulations by "sketching" their input parameters directly onto the rendering.

2.5 Chapter Conclusions

Multiple simulations with perturbed input parameters are performed in many application domains, such as engineering and geosciences. Analyzing the results of simulations can help us understand the behavior of the system. Comparing the system's response to different parameter values enables us to understand its sensitivity to the parameters, and optimize the parameters such, that the desired response is achieved. The data sets produced by those repeated simulations (as well as repeated measurements) exhibit several aspects of complexity: they are multi-dimensional, time-dependent, multi-run, and often have spatial context. The visual analysis of such data sets is particularly difficult, because each aspect can pose different challenges [109, 118]. Visual analysis tools need to usefully tackle all of those aspects.

In this chapter we provided an overview of the related state of the art. There are many visualization and visual analysis approaches that address some of those aspects of complexity, but few handle all of them. One popular technique to fuse different approaches is coordinated multiple views [118, 214], where individual views show different perspectives of the data. Multiple attribute dimensions (including simulation input parameters) can be displayed in well-known information visualization views [166]. Spatio-temporal data can be displayed, for instance, using volume rendering [6]. Statistical aggregates of multi-run data can be visualized by embedding glyphs in the 3D visualization [120].

The challenge in such systems is to coordinate the views effectively, i.e., enable the user to correlate data items in the individual views, also the ones that represent different levels of abstraction [120]. The visual analysis system must provide intuitive and efficient means of interaction so that the user can select, explore, filter, and connect items in the visualization [288]. The interaction helps the users to relate his or her internal mental model of the analysis to the visualization [153, 236], so that they can formulate, verify or reject hypotheses.

It is very promising to integrate visual and automated analysis methods [121, 246]. Automated analysis can usually deliver quantitative, more accurate, and more reliable answers. It is generally faster and cheaper, and it scales also to very large data sets. However, it requires that the problem is precisely defined. It may not work well for new, unanticipated types of analysis tasks. Conversely, visual analysis can tackle vaguely defined problems, and incorporate domain knowledge and human intuition. The interplay of the two approaches constitutes a very powerful analysis methodology. In an iterative process, the insights gained via one approach can help formulate hypotheses also for the other one. Similarly, hypotheses generated by any of the two approaches can be confirmed or rejected also by the other one, and lead to new insights. Through this cooperation, data analysis problems can be solved that are not solvable with neither of the individual approaches [129].

The ultimate goal of visual analysis is to support domain experts in generating knowledge from data. That said, it is interesting to observe that the communication between the visualization community and application domains leaves a lot to be desired [282]. Domain experts rarely use recent results in visualization, and visualization research rarely incorporates the knowledge of domain experts in designing new visualizations and analysis tools. Johnson [109] suggests that establishing mutually beneficial peer relationships between visualization and application experts is one of the most important factors in increasing the acceptance of visualization, and ultimately integrating it with the analysis workflow. Recently, Sedlmair et al. [232] survey the design of problem-driven visualizations.

Chapter 3

Interactive Visual Analysis of Families of Function Graphs

"It is only in appearance that time is a river. It is rather a vast landscape and it is the eye of the beholder that moves."

— Thornton Niven Wilder (1897–1975)[1]

The exploration and analysis of multidimensional and multivariate data is one of the most challenging areas in the field of interactive visual analysis. In this chapter, we describe an approach to the visual analysis of an especially interesting set of problems that exhibit a complex internal data structure. We describe the interactive visual exploration and analysis of data that includes several families of function graphs $f_i(\mathbf{x}, t)$. We describe a data model that handles function graphs as atomic data types. We discuss analysis procedures and practical aspects of the interactive visual analysis with special emphasis on the function graph characteristic of the data. We adopted the well-proven approach of multiple, linked views [214]. Standard views, such as histograms, scatter plots, and parallel coordinates, as well as the function graph view are used to jointly visualize data. Iterative visual analysis is supported by providing means to create complex, composite brushes that span multiple views and that are constructed using different combination schemes.

This chapter is organized as follows. Section 3.1 provides context and motivation. Section 3.2 gives a brief description of the data model used and the exploration procedures. Section 3.3 describes our proposed tools and methods for supporting these tasks. Section 3.4 introduces the typical tasks in the analysis of such data sets. Section 3.5 contains closing remarks. A case study of the optimization of a Diesel fuel injection system based on the concepts in this chapter is described in Section 7.1.

[1] American playwright and novelist, three time Pulitzer Prize winner.

3.1 Motivation

The development of effective visualization and interaction techniques requires the understanding of the properties of the data and the typical tasks the users want to perform [246]. Unfortunately, this requirement is not always met, often because of insufficient collaboration and communication between visualization experts and the users. The users' ultimate goal is to find expected phenomena to support (or reject) their hypotheses or to discover unexpected results that question their assumptions or the validity of the data acquisition process. That can lead to the generation of new hypotheses.

The challenges of data analysis and exploration are associated with large and complex data sets, increased dimensionality, and the consideration of data semantics, including features, focus and context [59]. Therefore, a visualization tool should be designed in close collaboration with potential users. Tool developers must be aware of the users' actual requirements, the usual tasks they need to solve, the shortcomings of their previously used tools, and their feedback on new ideas. A part of that process is a development of intuitive and effective visualization and interaction techniques based on a common data model. If designed well, the same principles can be used across several application domains, including engineering design [138, 139, 164, 161, 163, 166], medicine [160], and ethology [168].

Modern simulation software can generate massive amounts of complex data that require suitable analysis techniques to get an insight into the practical implications of the results. Simulation is increasingly used to assess the quality and potential of new designs early in the process for example in aeronautical design [17, 65] or in the automotive industry [163]. Building real prototypes is time-consuming and expensive. Even though measurements on test bed systems are likely to remain an important way to verify designs in the future, the use of computational simulation in the design and production process can help to minimize the costs of the development and shorten the time-to-market for new products.

For example, in the automotive industry many different aspects of new designs are checked using simulation long before a new car is manufactured. Examples include mixture formation and combustion, engine cooling and filter regeneration, air conditioning in the passenger cabin and front shield deicing, and many others. The increasing complexity of automotive subsystems, e.g., the power train, the intake and exhaust system, or the fuel injection subsystem, also requires simulation for optimization. Tuning of an injection system for modern cars is an example of a multi-parameter optimization process. The operation of the injection system depends, in a very indirect way, on several parameters. Therefore, optimization by experience and/or intuition is usually not possible.

In this chapter, we present a new approach to the interactive visual exploration and analysis of simulation (or measurement) data containing families of function graphs. This approach is general enough for a number of application scenarios that share the same characteristics, including multi-parameter tuning problems. A major challenge, in general, is how to visually relate the multivariate dependent variables to their multidimensional reference parameters (independent variables). We suggest a combination of different kinds of views with specific brushing interactions, all adapted to work well for the families of function graphs in order to facilitate the interactive visual exploration and analysis of such data sets. We introduce our concept via

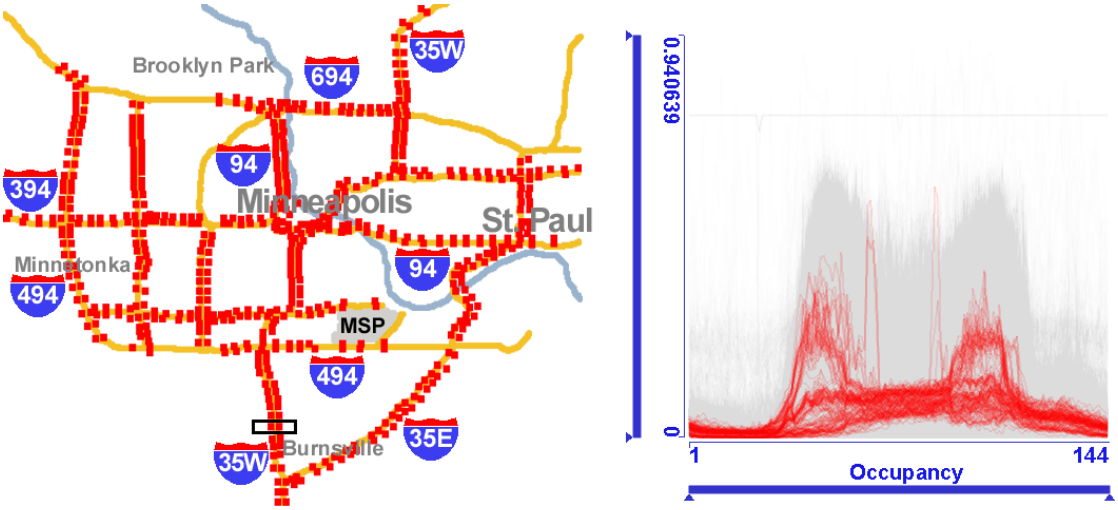


Figure 3.1: Left: the map of all sensor locations in the Minneapolis freeway system traffic data. A schematic map is underlayed to provide context information. Each red dot marks the location of one *station* (which usually encompasses several sensors, *detectors*, one per lane). Right: road traffic occupancy family of function graphs. Data from some sensors (marked with the black rectangle on road 35W in the map) is highlighted in red. Occupancy is defined as a percentage of time a detector detects vehicles. It is measured in ten minute intervals. An occupancy value of 0.7 means that for seven out of ten minutes a sensor detected vehicles.

an illustrative example analysis of road traffic measurement data. The case study of a Diesel fuel injection system optimization in Section 7.1 demonstrates the usability of our ideas in a real-world engineering context.

3.2 Data Model

Generally speaking, a data model consists of a data definition and a manipulation language (structuring and operational definitions) [248]. Data definitions that result from engineering simulations, real-world sensor data sets, or intelligence data may be very similar. Consequently, the data sets under consideration share some common characteristics.

3.2.1 Data Definition

The data sets contain values for m independent variables and n dependent variables. The independent variables $\mathbf{x} = [x_1, \ldots, x_m]$ are real-valued and their values define a subset I of the data set. A member of $I \subseteq \Re^m$ represents a specific set of values $\mathbf{x}_i$ of independent variables. For each $\mathbf{x}_i$, the corresponding set of values of dependent variables is provided; this data model does not handle missing values. There are two types of dependent variables, regular and function graphs. While regular variables have a singular value for each $\mathbf{x}_i$, function graph variables use an additional independent variable (often time) to provide a set of values for each $\mathbf{x}_i$. A

(a)

Record #	TimeStep	Day	SensorX	SensorY	Volume	Occupancy
1	1	1	167.125	-637.000	0.432	0.114
2	2	1	167.125	-637.000	0.412	0.132
3	3	1	167.125	-637.000	0.368	0.157
...	...	...	...	...	...	...
144	144	1	167.125	-637.000	0.451	0.092
145	1	1	1534.375	1883.000	0.289	0.083
...	...	...	...	...	...	...
289	1	1	1709.125	3760.000	0.154	0.129
...	...	...	...	...	...	...
8064000	144	28	159.250	175.000	0.216	0.167

(b)

Record #	Day	SensorX	SensorY	Volume(t)	Occupancy(t)
1	1	167.125	-637.000		
2	1	1534.375	1883.000		
3	1	1709.125	3760.000		
...	...	...	...	...	...
56000	28	159.250	175.000		

Figure 3.2: Two approaches to the management of function graph data. (a) The independent variable of the function graphs (e.g. time) is represented by adding an additional dimension. Records 1 to 144 represent one function graph. The scalar dimensions of the records need to be duplicated for each value of the independent variable. (b) A function graph is represented as an atomic type in the data. Duplication of scalar dimensions is not necessary. Columns of this table are families of function graphs.

function graph can be visualized as a 2D plot that shows how the value of a dependent variable changes over time. In other words, the regular variables $\mathbf{r} = [r_1, \ldots, r_{n_r}]$ depend only on $\mathbf{x}$ while the function graph variables $\mathbf{f} = [f_1, \ldots, f_{n_f}]$ depend on $\mathbf{x}$ and time $t \in \Re$. For a specific set of values $\mathbf{x}_i$ of independent variables and fixed time t_j we can define the set of values of dependent variables as $\mathbf{d} = [r_1(\mathbf{x}_i), \ldots, r_{n_r}(\mathbf{x}_i), f_1(\mathbf{x}_i, t_j), \ldots, f_{n_f}(\mathbf{x}_i, t_j)]$, $n_r + n_f = n$. The dependent variables and their values (possibly, over time) define a subset D of the data set. For a given function graph variable, $f_j(\mathbf{x}, t)$, we define a *family of function graphs* as a set of function graphs for each possible value of $\mathbf{x}$, $\{f_j(\mathbf{x}_i, t) | \forall \mathbf{x}_i \in I\}$.

We use a road traffic measurements data set to illustrate the concepts described in Sections 3.3 and 3.4. The data set is provided by the Traffic Management Center of Minnesota Department of Transportation[2] that maintains an archive database of road traffic measurements from the freeway system in the Twin Cities metropolitan area (Figure 3.1). The data set contains 28 days of measurements from approximately 4,000 sensors grouped into about 1,000 stations covering ten main roads in the Twin Cities metropolitan area. Opposite directions on a road (e.g., northbound vs. southbound) are treated separately, thus effectively creating 20 one-way roads. I consists of the positions of the sensors, road numbers, and weekdays. The sensors report traffic volume and occupancy, thus D consists of two families of function graphs in this data set. Each function graph in each family represents one day's worth of measurement data, so there are in total about 112,000 function graphs in the data set (4,000 sensors times 28 days). The sensor data are aggregated into 10 minute intervals, therefore each function graph contains 144 points to represent the 24 hours of a day.

Such data in databases and also in IVA applications are generally stored as records that consist of attributes. This concept is well-known, records can be considered as points in a space

[2]Twin Cities Traffic Data Archive, `http://www.d.umn.edu/~tkwon/TMCdata/TMCarchive.html`

of $m + n$ dimensions, representing all (independent and dependent) attributes. All attributes (dimensions) are scalars, either numeric or categorical. We omit a more detailed discussion including nominal and ordinal types here. Time-dependent data like function graphs can be represented by adding time as an additional dimension.

The table in Figure 3.2(a) illustrates a conventional way of storing such data. Time appears as an additional column in the table. For each combination of the independent variables (day and sensor location), there are 144 records to represent a function graph variable, one record for each time step. The scalar dimensions of the records need to be duplicated for each time step. If, however, we take into account the nature of the data and allow some columns to contain function graphs, we get a different model, shown in Figure 3.2(b). We have now one record that contains the time series *Volume(t)* as a function graph, as opposed to having 144 records with *TimeStep* and *Volume* attributes, as in Figure 3.2(a). Duplication of scalar dimensions is not necessary. Note that we have substantially reduced the number of data records and, simultaneously, increased the complexity of the data model. We did not lose any of the data; to the contrary, we have gained additional information. Now values pertaining to a given day and sensor location are grouped into a single record. All function graphs populating one column in Figure 3.2(b) constitute a *family of function graphs*.

If dimensions can be not only scalars, but also function graphs, then we can improve the analysis significantly. This data model has proven itself in several case studies we have done in different fields, for example optimization of Diesel fuel injection (Section 7.1), timing chain drive design (Section 7.2), medical data [160], and ethology [168]. Fang et al. [67] have proposed a similar data model for medical image data sets.

3.2.2 Manipulation Language

Once the data set is defined, the question is how to analyze the data. In our data model, the manipulation language is an exploration language that enables search and pattern discovery without modifying the data set. From the visual analytics point of view, the goal is to discover, in an iterative manner, trends, tendencies and outliers in the data and to see how patterns in D map to the corresponding subsets in I and vice-versa. In order to achieve that, data exploration techniques must be conceptually simple, easily combined and visually intuitive.

The visualization framework is based on the described data model and a set of visual operators (brushing techniques) and views (histograms, scatter plots, parallel coordinates, etc.) that are linked together. The design of interactive visual analysis within this framework is based on the following principles. The analyst can select a varying number of views. Within each view, the variables of interest can be selected and the corresponding values displayed. The visual operators are used to select a subset of "interesting" values for the specific variables in the view. The selection is immediately displayed in all other views. Families of function graphs are of special importance in providing a visual space for patterns. Within a family of function graphs, we would like to select function graphs based on their shapes. It is possible to use a combination of function graph values to specify the desired shape of a function graph, i.e., the pattern.

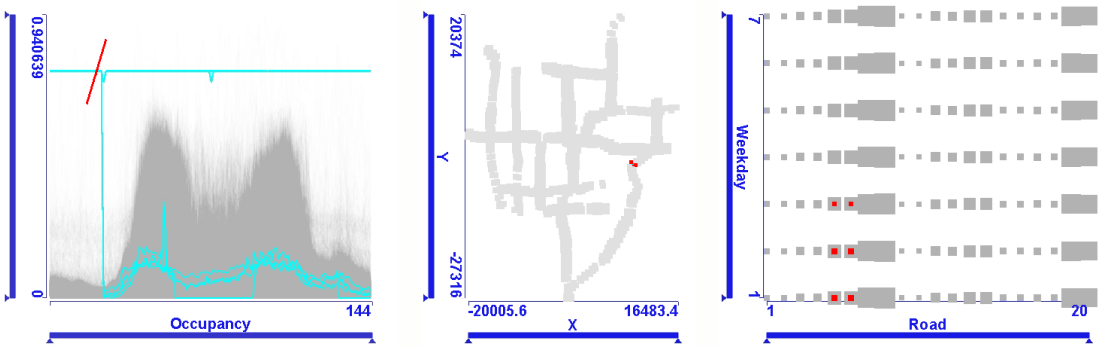


Figure 3.3: Several occupancy function graphs of atypical shape have been selected by the red line brush. We conclude from very high occupancy values that those function graphs indicate malfunctioning sensors. In the linked map view (scatter plot view of sensor coordinates) we can see that there are two malfunctioning sensors next to each other. In another linked scatter plot view weekdays and road numbers are displayed. Each column represents one direction (for instance, southbound) of a road. We can see that those sensors are on road 35E and that they did not work for three days.

3.3 Tools for the Analysis of Families of Function Graphs

We have developed a tool based on premises described in Section 3.2. The combination of basic, highly interactive views is sufficient to carry out a wide range of sophisticated analysis tasks. Interactivity plays a crucial role in analysis. Important and novel aspects that support interactive procedures are described in the following.

We currently offer a combination of linked views including histograms, scatter plots, parallel coordinates and function graphs. We do not make any assumptions about independent and dependent variables in the sense that we would restrict any of the basic views to display either of them. The views can display any attribute of the data set, both independent and dependent variables. The user can create new views, arrange views as desired, and can have more than one instance of the same view type showing the same or different attribute sets. In practice, display sizes typically limit the number of views to six. It is possible to temporarily maximize one view for more detailed examinations. Histograms, parallel coordinates and scatter plots are standard, well-known views [40], thus we do not describe them in detail. However, it is worth mentioning that the point size in scatter plot views can optionally be made proportional to the number of data items represented by a single point (compare to Piringer et al. [195]). The more items a point represents the larger it is. An example is shown in Figure 3.3: larger points indicate more sensors on the road. Similarly, the sizes of points highlighted in the focus set are also proportional to the number of items brushed. Thus the ratio of brushed items versus context represented by a point in the scatter plot is indicated by point sizes.

The function graph view displays a family of function graphs at once. If the number of function graphs in the family is large then the display can become visually cluttered and non-informative. In order to represent the characteristics of the family better, we can (optionally)

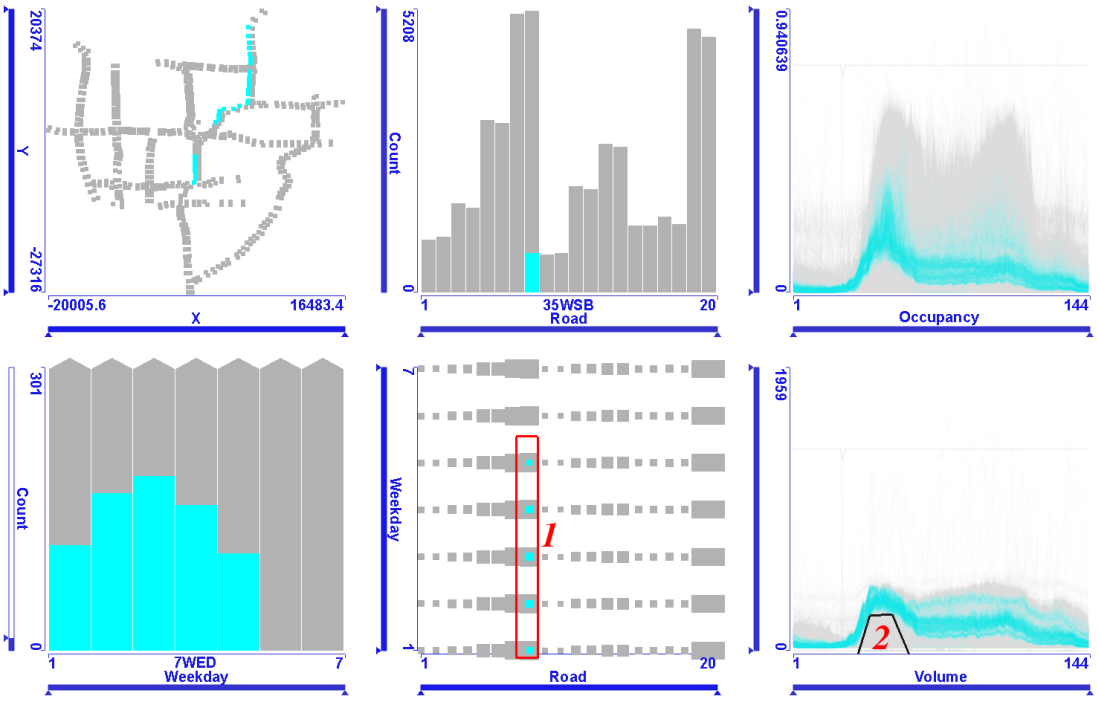


Figure 3.4: A snapshot from an interactive visual analysis session of traffic data in the Minneapolis metropolitan area. Red numbers indicate the brush creation order. Here we look for locations of high volume morning traffic on a given road on weekdays. The user has selected the road and weekdays in the scatter plot (brush 1) and then removed low traffic volume graphs intersecting brush 2 using SUB operation. The locations are highlighted in the linked map. The road number (35WSB, i.e., 35W southbound) is shown on mouse-over. We can see that heavy morning traffic on southbound 35W occurs mainly north of the downtown area, i.e., towards it, and towards the airport, which is south of downtown Minneapolis.

render the pixels where more graphs pass through with higher intensity, effectively producing an alpha-blended display.

In the following we describe generic interaction principles and elaborate on the specific requirements of brushing and linking in various views.

3.3.1 Generic Interaction Features

If the basic views listed above are independent, then they provide a limited insight into the data set. However, if areas of focus can be highlighted with applicable brushing techniques and this focus area is linked to the other views, then correlations and dependencies in the data can be revealed. Our system supports interactive brushing and linking (Figure 3.4). The number of currently brushed data items versus the total number of items is always indicated. The user can perform brushing in any of the views and all the other views will highlight the brushed items

while the context is shown in a different, less saturated color. Whenever applicable, the view can be zoomed to show the brushed region only. The brushes can be resized and dragged to new locations, which helps in the interactive data exploration. A tabular display of the currently brushed items can be opened when the user needs detailed numeric information.

With simple brushing and linking it is usually a problem to locate the matching brushed data items in different views. If more data items are brushed in a view then all corresponding items are highlighted in other views, but we cannot visually identify the same item in the different views. We have applied an optional *color gradient* along the brush and used this color gradient in the linked views to establish a visual link between the correlated data items. This also helps in discovering tendencies in the data set. Figure 7.7 shows an example where the color gradient establishes visual links between a scatter plot and the function graph view.

Another improvement is *composite brushing*, a query tool which is a result of logical operations performed on brushes. Composite brushing makes it possible to build queries that specify several overlapping or intersecting ranges of criteria in the same or different views. We could have chosen to offer AND, OR and XOR operations to composite brushes and add a formula editor to allow controlling the order of operations by bracketing, similar to the *feature definition language* proposed by Doleisch et al. [59]. In contrast, we use composite brushing by offering AND, OR, and SUB operations where the first operand is always the result of the latest composition. This allows a simplified, intuitive, and more iterative workflow compared to working with a formula editor. The user defines the first brush, then (optionally) selects a Boolean operation, adjusts the composition setting (to either AND, OR, or SUB) and then defines the next brush to adjust the current selection. Subsequent brushes and operations will be applied to the result of prior brushes only. Iteratively, every new brush alters the current selection status according to the composition rule in use. The process continues in this way: new brushes and operations are applied to the latest state only. Each new brushing operation provides immediate visual feedback and the user can interactively refine (using AND and SUB) or broaden (using OR) the current selection and steer the information drill-down. The user can also resize or move any existing brush in the chain to gain even more flexibility.

Brushing and linking is a powerful feature in understanding how outputs depend on inputs and finding input parameter sets when desired properties of outputs are known. Because we treat all variables in the same manner, one can brush in views of independent variables and study how dependent variables change in other views, or perform the inverse kind of investigation to find suitable inputs for specified results by brushing in the views showing output parameters.

Brushing conventional views is quite straightforward [158]. The user can select histogram bins, rectangular areas in scatter plot views or ranges of parallel coordinates axes. We have introduced a novel brushing tool in the function graph view that we describe in more detail.

3.3.2 Brushing Function Graphs

We suggest two brushing methods to meet the specific requirements of queries on families of function graphs.

A *line brush* is a simple line segment drawn in the function graph view. It selects all function graphs that intersect the line. Figure 3.3 shows an example of selecting several graphs that have high and constant occupancy value. That indicates malfunctioning sensors. Linking them to

the corresponding points in I in the map, we identify those sensors. It is very easy to exclude outliers in a family of function graphs or to isolate function graphs with desired characteristics with just a few line brushes (see brush 2 in Figure 3.4). Additionally, it is very useful to provide a polyline brushing opportunity, i.e., a brush in the form of a polyline which selects all function graphs which intersect any of the polyline segments. The line brush, together with the above-mentioned composition functionality, assists the user when looking for function graphs whose approximate shape is known. A logical operation can be defined individually for each line brush which supports very complex queries. An example of composite brushing is shown in Figure 3.6. A complex combination of line brushes is used to include and remove various function graph shapes in the focus set. We have found compositions of line brushes very intuitive and effective in brushing function graphs.

A *rectangular brush* selects all function graphs that pass through the rectangle. The time-box widget [96] is an analogy to rectangular brushing. We have enhanced the original idea by allowing the user to optionally limit the brushing to function graphs that enter and leave the rectangular brush at given edges. Probably the most useful ones are those where the function graph is required to enter and leave at the bottom or on the top edge of the rectangle. These function graphs have a local maximum or minimum inside the rectangle, which is often a criterion in the analysis of function graphs. This is especially useful if the display of a family of function graphs is dense and areas of maxima and minima are occluded by other function graphs. Alternatively, the similarity brush [35, 96, 178] (see also Section 4.3.3) and working with the first derivatives of the function graphs (see Section 4.3.2) can also be useful in such cases. The rectangular brush can be represented as a composition of line brushes.

3.4 Analysis Procedures

The shapes of function graphs depend on the independent variables $\mathbf{x}$ and in practical cases the shapes usually exhibit similarities for slight variations in the variable values, albeit this correlation may be quite indirect. For example, complex physical systems can be considered "black boxes" that return output for an input parameter set, but their exact dependencies on the inputs are unknown. This can also happen if a system is simulated using a computer. In that case the boundary conditions can have so diverse effects on the results that in the analysis of such systems it is more feasible to reconstruct the black box by exploration rather than by trying to deduce its internals from a simulation process. The analysis and exploration of this class of data involves several types of procedures, including discovering trends and tendencies or finding outliers in D. For certain data sets, similar analysis of I can also be of interest. However, in this section we focus on finding patterns and dependencies in the union of I and D.

3.4.1 Black Box Reconstruction

We call the process of understanding the influence of independent variables on dependent function graph variables *black box reconstruction*. To accomplish this, we usually need to have an overview of the entire data set, following the principles of Shneiderman's visual information seeking mantra: overview first, zoom and filter, then details-on-demand [236]. We are interested

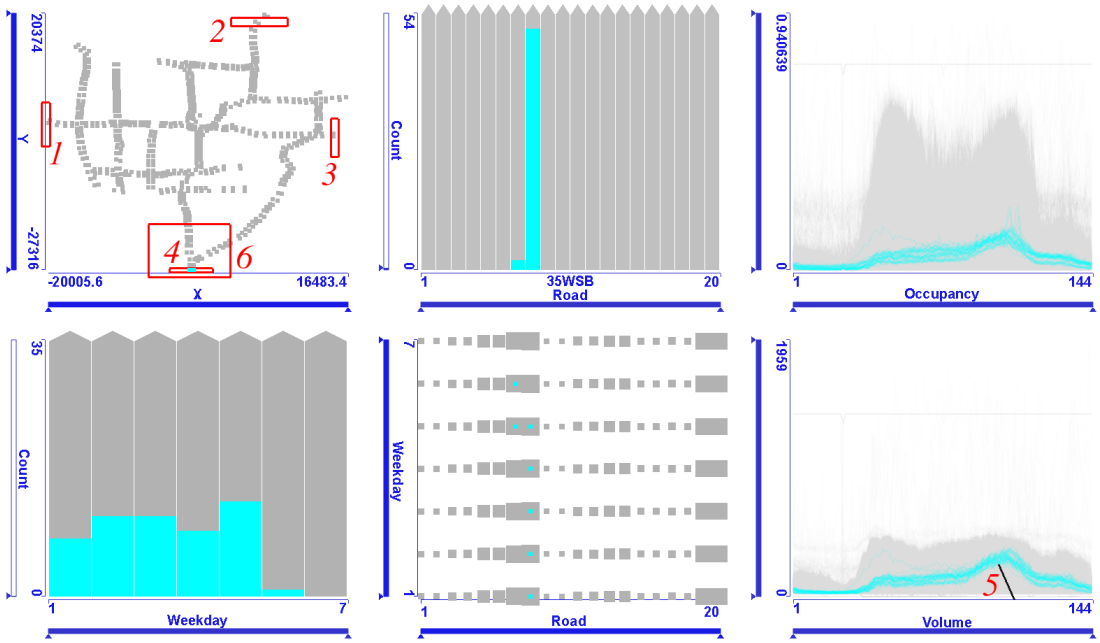


Figure 3.5: Another snapshot of an iterative, interactive visual analysis of the Minneapolis traffic data. Red numbers indicate the brush creation order. First, entries into the freeway system are selected using the union (OR) of four brushes (labeled 1 to 4) in the top left map. Next, low volume evening traffic is excluded from the focus by subtracting graphs which intersect brush 5. Using logical AND with brush 6 in the map view we can restrict the investigation to one specific entry. By dragging this brush to other entries we can quickly change the focus to one of the four entries. Thereby a comparison of traffic patterns with respect to the four entries is possible. In this snapshot evening traffic on the south entry is shown. The highlighted points in the lower middle scatter plot and the linked histograms reveal that heavy traffic direction is southbound from Mondays to Thursdays, but interestingly, shifts to northbound on Fridays and Saturdays (road number and direction are displayed on mouse-over).

in how dependent variables, including function graphs, change as values of independent variables are changed. We want to fix values of some independent variables to reduce the focus area and vary other independent variables while studying the corresponding dependent variables and function graphs. This is an interactive and iterative data exploration process: brushes are created and moved to areas of interest. When we have built up an overview of the dependencies we want to zoom in on details in both I and D in order to discover more subtle correlations in the data. In case of function graphs it is especially important to provide context information so that changes in shape are more obvious as various ranges of values are brushed.

An example of black box reconstruction is shown in Figure 3.5. We are interested in evening traffic characteristics entering the Minneapolis area freeway system. The freeway system has entrances from North, South, East and West. We first brush these entry points in the map view with a logical OR combination of four brushes labeled 1 through 4. Then low traffic volume in the

evening is excluded using a line brush with SUB operation (5). Finally, we create a larger brush in the map using the AND operation (6) to restrict the focus to one of the entries. By dragging this last brush to the other three entries we can quickly change the focus and compare traffic patterns of the four entry points, while still being accurate with respect to brushes 1 to 4. After each interaction step all views are immediately updated in order to support iterative analysis.

3.4.2 Analysis of Families of Function Graphs

In this type of analysis we (approximately) know the desired or expected shape of function graphs and our goal is to find combinations of independent variables that produce those shapes. We also want to exclude combinations that produce undesirable or invalid function graphs and we want to find out how the deviations from the desired shape depend on the independent variables. This could be considered an inverse investigation compared to the one in Section 3.4.1. However, this type of analysis requires that we have an idea of the operation of the black box so that we avoid erroneously identifying dependencies that are a mere coincidence.

The procedure requires focus+context views of the graphs where criteria can be defined to select graphs of specific shapes. The desired shapes of graphs can be characterized by brushing. The typical procedure is to locate invalid or undesired function graphs first, as illustrated in Figure 3.3. We brush them in the function graph view and find the related values of independent variables, in this case locations of the malfunctioning sensors. We will exclude these items from further analysis.

The desired properties of a function graph can be defined by line brushes, as shown in Figure 3.4. Here we look for locations of high volume morning traffic on a given road on weekdays. This can be accomplished by selecting the road and weekdays in the scatter plot (brush 1) and then removing low traffic volume function graphs using SUB operation (brush 2). The locations are highlighted in the linked map.

3.4.3 Multidimensional Relations

Another interesting aspect of the analysis is the correlation between different families of function graphs. We want to investigate features of one family of function graphs depending on the properties of a set of function graphs in another family, for example, relationships between traffic volume and occupancy. This analysis within multidimensional time series data requires that families of function graphs are displayed simultaneously and the user can interactively brush specific groups of function graphs in one family and study the corresponding ones in the other families. We may also want to narrow the search by specifying filters on the function graph's independent variables $\mathbf{x}$. Furthermore, we want to be able to define criteria for various families of function graphs.

As an illustrative example let us consider the following query on the traffic data: we look for areas where traffic is strong, but still moving both in the morning and in the afternoon. Those are heavily used roads without traffic jams. If cars travel at higher speeds then many cars pass over the sensors but with relatively large gaps. This is indicated by high volume and relatively low occupancy values. In the analysis tool this is expressed as a combination of brushes in both families of function graphs. There is no direct correlation between the two function graphs. The

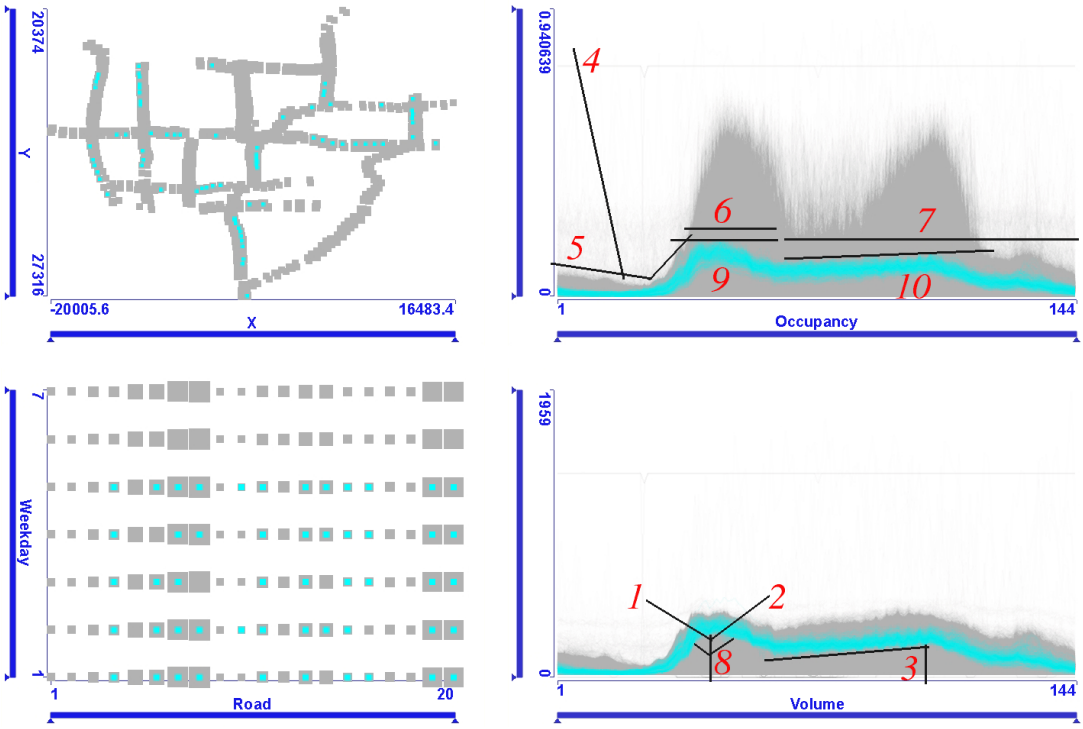


Figure 3.6: We look for roads where traffic is high both in the morning and in the afternoon. First we select heavy morning traffic (brush 1 AND brush 2), then remove low evening traffic volume (SUB 3). Data from malfunctioning sensors is removed (SUB 4). Next we remove high occupancy function graphs by subtracting brushes 5, 6 and 7. The desired function graph shapes are further refined by removing function graphs which intersect brushes 8, 9 and 10. The combined criteria on the two function graphs reveal the roads in question on the map. We can also see in the lower scatter plot view that interestingly, on specific roads, no points are highlighted on Wednesdays. This means traffic on those roads does not follow this pattern.

investigation is demonstrated in Figure 3.6. We brush in the volume and occupancy function graph views and study the linked map. First large morning traffic volume is brushed using two line brushes (1 AND 2). Then we remove function graphs with low traffic volume in the evening (SUB 3). Now we narrow down the search in the occupancy function graph view. Data from malfunctioning sensors is excluded (SUB 4). We then limit the occupancy by removing function graphs which intersect brushes 5, 6 and 7. Now we have a view of the areas where traffic is strong but moving in the morning and in the evening. We can further refine the desired traffic volume shape in the morning by removing the function graphs which intersect the three line brushes (SUB 8). Finally, we limit the occupancy to even lower ranges by removing function graphs that intersect brushes 9 and 10.

The highlighted points in the map in Figure 3.6 show those roads. We can also see correlations between days of week and traffic volume. Points corresponding to weekends are not

selected, as expected. We can also see something unexpected: on some roads, no points are highlighted on Wednesdays. On those roads, traffic does not follow the above pattern on Wednesdays. The road number (road 35 westbound) is displayed on mouseover.

3.4.4 Hypothesis Generations via Visual Analysis

There is a particularly strong need in engineering applications to perform automatic optimization of designs using several simulation iterations with suitably varied boundary conditions. The automatic optimization process must have an approximate model of the simulation in order to know how boundary conditions should be adjusted in search for an optimum. Visual analysis can be used to create hypotheses and rules that the automatic optimization can use in its simplified model and also to find out if the optimization misses some families of function graphs while searching for an optimum.

Gaining insight into the current design and setting up hypotheses about its operation via visual analysis has a very important additional advantage over pure numeric optimization. When designing a new component, engineers almost never start from scratch, but the new design evolves from an old one. Because of this iterative nature of design in engineering, the insight gained from the analysis of previous designs can be useful in improving future ones. This also implies that the simulation models of new designs are not radically different from those of the old ones and their results are comparable to some extent, too. By analyzing the relationships between the two, tendencies can be found and extrapolated to improve future designs.

3.5 Chapter Conclusions

The analysis of relationships between families of function graphs is a common task in many application domains. A novel combination of established visualization techniques, linked views and advanced brushing features represents a valuable tool for interactive visual exploration and analysis of data sets that include families of function graphs. Independent and dependent variables in the data set are treated the same, providing improved support for iterative exploration and analysis of the entire data space. Multiple, linked views enable simultaneous viewing of independent and dependent variables with immediate feedback.

Brushing proved to be especially effective, since it allows the interactive exploration of relations between independent and dependent variables. The color gradient improves the visual connection of the brushed items to the linked focus+context visualizations. The composite brushing with AND, OR, and SUB operations supports the iterative refinement of information drill-down and the detection or extraction of patterns from the application domain. The line brush technique proves to be especially useful in selecting function graphs. It is intuitive, easy to use and very effective. Figure 3.6 shows how a composition of nearly a dozen line brushes is used to identify a pattern in a family of traffic volume function graphs.

The process of the composite brush construction captures the essence of visual analytics procedures: it is interactive and iterative. The initial brush provides the initial data selection in one view. That selection is immediately displayed in the linked views and analyzed from different perspectives to formulate a hypothesis that is then tested using new brushes. During this iterative

procedure new, possibly unexpected patterns can be found. Figure 3.3 shows a discovery of a pattern in D (constant high occupancy) that indicates a pattern in I (malfunctioning sensors). Such discoveries are more difficult or even impossible without interactive visual analysis.

Chapter 4

Analysis using Data Aggregation and Derivation

"The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them."

— Sir William Henry Bragg (1862–1942)[1]

Time-series data are regularly collected and analyzed in a wide range of (scientific) domains. Acquiring values of a physical quantity via simulation or measurement over time produces time-series data. The data can be described as a function of time and it can be represented as a function graph. Multiple simulation runs or multiple measurements of the same physical quantity result in ensembles of function graphs which we call *families of function graphs*. The analysis of function graphs, (more generally: curves) is extensively studied in mathematics, statistics, and visualization; but less research is focused on the analysis of entire families of function graphs. Interactive visual analysis in combination with a complex data model, which supports families of function graphs in addition to scalar parameters, represents a premium methodology for such an analysis. In this chapter we describe the three levels of complexity of interactive visual analysis we identified during several case studies from different disciplines. The first two levels represent the current state of the art. The newly introduced third level makes an in-depth analysis of families of function graphs possible. The two essential components of the third level analysis are data derivation and advanced interaction. This is often the only way to extract deeply hidden implicit information from complex data sets. We seamlessly integrate data derivation, advanced interaction, and visual exploration to facilitate interactive visual analysis of families of function graphs. We illustrate the proposed approach with typical analysis patterns identified in two case studies from the automotive industry.

[1] British physicist, chemist, mathematician, and sportsman. Received the 1915 Nobel Prize in Physics for the analysis of crystal structure using X-ray diffraction.

4.1 Motivation

Generating useful knowledge from the information that is often only implicitly available in complex data sets is one of the key challenges in analysis. Interactive visual analysis (IVA) has proven itself, recently, as an important and valuable method of getting insight into, understanding, and analyzing complex data. However, IVA is still not well integrated into the whole analysis work flow. There is a multitude of powerful methods presented, but, unfortunately, they often remain isolated.

Our work is motivated by case studies we have done with domain experts from different fields including engineering [138, 139, 161] (see also Chapter 7) and medicine [160]. We have realized that many details of the data sets from those very different problem domains can be represented as *families of function graphs*, introduced in Section 3.2. We could also identify similar procedures in the analysis of such data sets. We suggest that representing function graphs as an atomic type opens new analysis possibilities. We focus on a set of tools for the analysis of data that contains families of function graphs.

The analysis of function graphs is a well-known and extensively researched topic in science and mathematics. However, there is not much research on the analysis of entire families of function graphs. As demonstrated in Chapter 3, IVA offers an effective and efficient opportunity to analyze even larger families of function graphs. The state of the art follows the visual information seeking mantra [236]: *overview first, zoom and filter, then details-on-demand*. This approach has often been proven powerful, but especially when working with families of function graphs, there are cases when zooming and filtering is not sufficient. Our work is in part motivated by curve sketching, a topic all of us are familiar with from high school. We need functionality to extend the data set by computing new attributes and additional derived curves when analyzing families of function graphs; just as we compute new attributes (e.g., extrema) and curves (e.g., first derivative) in curve sketching. Curve sketching helps us in the analysis of single curves. In this chapter, we propose related analysis methods for entire families of function graphs. The proposed ideas are in part suitable for the analysis of families of generic curves, too.

Traditional visualization systems often limit data manipulation to filtering and require more complex pre-processing to be performed in a separate step *before* the visual analysis. When the pre-processing is designed, one needs to estimate what properties of the data are expected to be of interest. Such a priori knowledge is often not available, in particular not in more intricate analysis cases. In this chapter we present a set of tools which extends the conventional approach and facilitates on demand data generation. By allowing the synthetic extension of data by attribute derivation (in addition to filtering, of course) at any time, new analysis possibilities arise. We claim that such an interactive attribute derivation mechanism is useful in particular for experts. An integrated system makes it possible to reveal deeply hidden information in the data without requiring detours or a priori knowledge to design the pre-processing.

The main contribution of this chapter is a set of analysis procedures, cleverly combined in one toolbox, which makes the deep and flexible analysis of complex data containing families of function graphs possible. We argue that a rich set of tightly integrated (general) analysis mechanisms can help in a wide range of application scenarios. We classify the analysis process

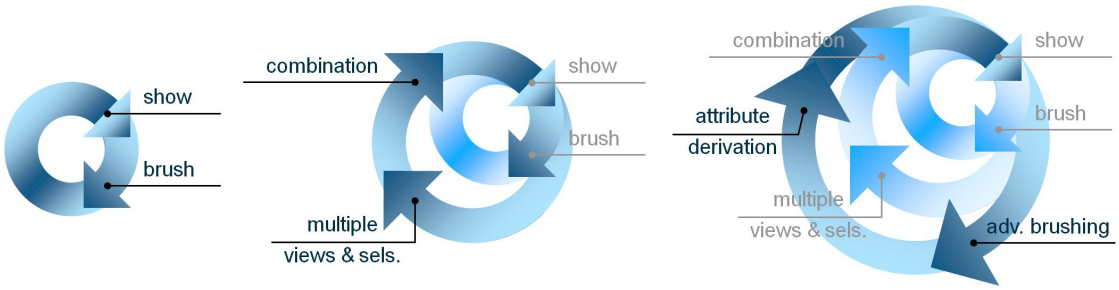


Figure 4.1: Interactive visual analysis on three levels of complexity. The first level is linking and brushing with one brush. On the second level, more views are used and brushes are combined with logical operators. On a third level, advanced interaction (brushing) and attribute derivation are added.

itself and suggest what should be included in an advanced IVA framework. We have identified three levels of complexity in the analysis process. The first two levels represent the current state of the art. We propose a third level that includes advanced interaction and on-demand data aggregation and attribute derivation in order to facilitate the exploration of deeply hidden details. We claim that there are two distinct ways to explore hidden features. One is to keep data intact and offer more complex interaction. The other possibility is to extend the data to be more complex and keep interaction simple. We demonstrate that the two approaches are complementary; there is no universally preferable way. We use examples from several case studies with domain experts from various fields to illustrate the possible use of the proposed combination of techniques.

4.2 Three Levels of Complexity in Interactive Visual Analysis

Interactive visual analysis is an iterative process. It usually starts with a simple analysis of the original data to gain overview. Then, for a more advanced analysis, the analyst needs to use more complex procedures and can combine findings from earlier stages of the analysis. The need for a more advanced analysis arises as information should be extracted which is more difficult to access. At a certain point it becomes difficult or even impossible to find features of interest by analyzing the original data only using conventional IVA methods. More advanced and complex interaction possibilities are required. Alternatively, or perhaps in addition to that, the data can be enhanced by computing aggregates or first derivatives of time series, for instance. Figure 4.1 illustrates our view on the three levels of complexity in interactive visual analysis.

We exemplify those three levels using a climate simulation data set [19]. Climate researchers try to develop models that can predict climate development in the future. The models are often tested and validated against known scenarios from the past. In our example, the simulated climate response to the outburst of meltwater from Lake Agassiz was studied [19]. The diffusivity parameters of the ocean model (two scalars, $diff_{\mathrm{H}}$ and $diff_{\mathrm{V}}$) were varied across multiple simulation runs. There were 10 variations of each parameter, producing *ensemble data* [118] of 100 individual simulation runs. In each simulation run, time series data are generated, including

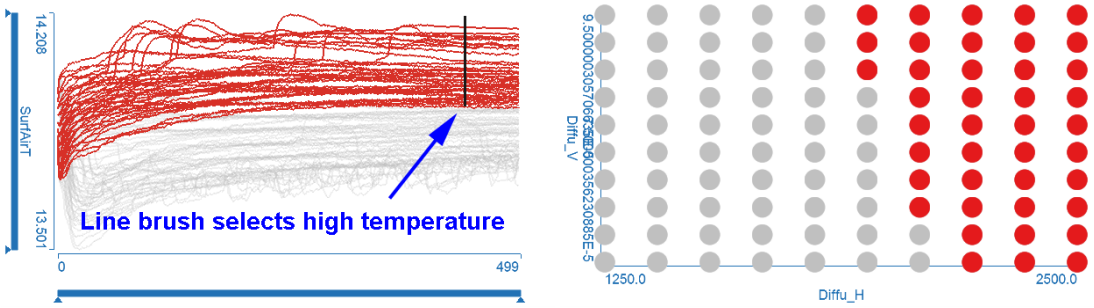


Figure 4.2: An example of a level one analysis with linking and brushing. The function graph view depicts air temperature. A line brush is used to select items with high temperatures at a later point in time. The corresponding model parameters $diff_{\mathrm{H}}$ and $diff_{\mathrm{V}}$ are highlighted in the scatter plot.

surface air temperature over land and oceans, global precipitation, and various differential measures such as heat transport and ice transport. Each simulation run spans 500 years, represented as 500 time steps in each series.

In a coordinated multiple views system, the *first level* (smallest circle, left in Figure 4.1) can be interpreted as simple linking and brushing with one brush. Simple brushing includes selecting a rectangular region in scatter plots, a range of an axis in parallel coordinates or using a line brush introduced in Section 3.3.2 in the function graph view. The user can interactively select some items in one view, and the selected data subsets are highlighted consistently in all views. Then a different set of items (another *feature of interest*) can be brushed and the highlighted patterns in the linked views can be studied. The user repeats this process, engaging in an iterative IVA loop. This is sufficient in many cases, for example, when identifying diffusivity parameters that lead to high temperatures (Figure 4.2), but we cannot formulate more complex queries. We cannot identify the droughtiest cases, for instance, because we would need to brush high temperature and low precipitation simultaneously.

On the *second level* (larger circle in the middle of Figure 4.1) more views are used and brushes can be combined with logical operators. This makes answering such questions possible, and it is at large the current state of the art [178]. Several brushes can be defined in the same or in different views. Brushes can be combined using logical operations (AND, OR, and NOT). Only the data selected by the composition of the brushes is highlighted in all views. Brushes can be composited via a *feature definition language* [59], or in an iterative manner, described in Section 3.3.1. This enables further drill-down (AND and NOT operations) or broadening (OR operation) of the selection. An example can be seen in Figure 4.3. The droughtiest cases can be found easily with composite brushing. For more detailed analysis of complex data this is still not sufficient. There are many more analysis goals that cannot be achieved by the use of those methods, for example, identifying cases which have rising temperature at some point in time, or finding temperature curves of some specific shape.

We can see some jagged lines in Figure 4.3, especially noticeable in the area marked with the green ellipse. There are also some faintly visible jagged lines in the precipitation plot. Is there

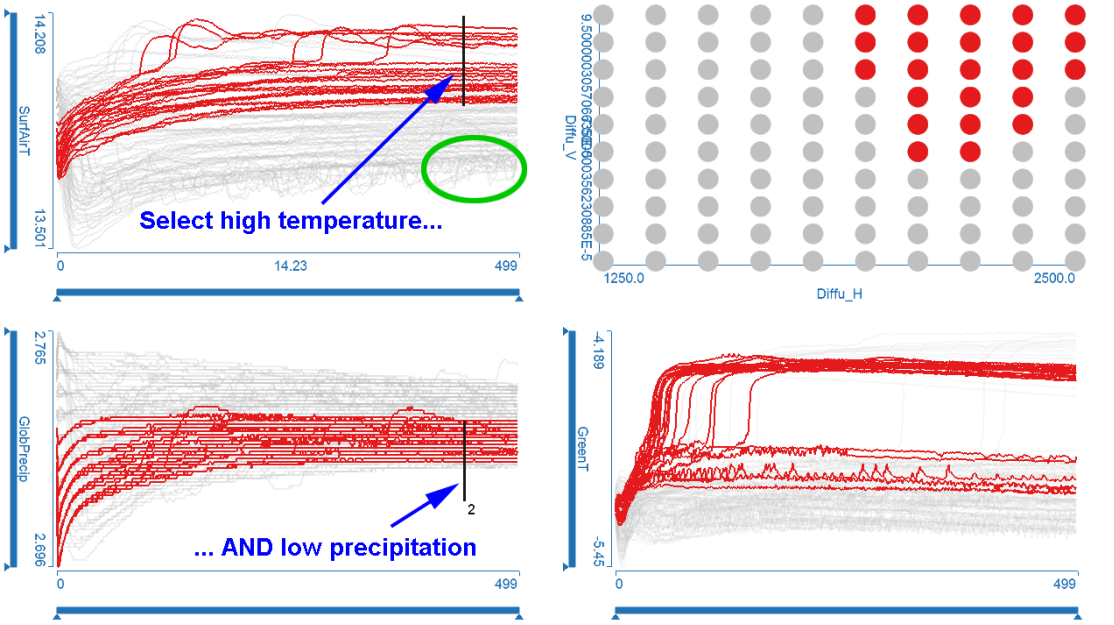


Figure 4.3: An example of a level two analysis. A composition of two brushes highlights the droughtiest cases. High temperatures (top left) and low precipitation levels (bottom left) at the same point in time are brushed. The scatter plot displays model parameters $diff_{\mathrm{H}}$ and $diff_{\mathrm{V}}$. Greenland temperature is shown in the bottom right. However, level two analysis is not powerful enough to select the jagged lines marked by the green ellipse.

any correlation between them? Do they belong to the same simulation run? How do we select them? One possibility is to compute the first derivatives of the function graphs. Extreme (either positive or negative) values of the first derivatives can be brushed to select jagged curves. The procedure is illustrated in Figure 4.4. It must be mentioned that the first derivative of discretely sampled data is approximated by finite differences, and the data often needs to be smoothed before differentiation to compensate for frequency amplification due to derivation.

We observed that there is a certain duality in complex analysis tasks. One option is to use complex interaction methods and visualizations, and retain the data set in its original form. Complex interactions include, for instance, the angular brush in parallel coordinates [89] and the angular line brush [161] in the function graph view. Conversely, new data can be synthesized during the analysis. This derived data can often be analyzed with simpler interaction techniques. Both approaches have advantages and disadvantages. There is usually a learning curve associated with complex visualization and interaction techniques before analysts can use them effectively [126]. On the other hand, the meaning of derived data is not always intuitive. Depending on the task and the analyst's experience and background one or the other method is preferred. Our experience is that none of the two approaches is universally preferable. The interactive visual analysis framework needs to provide support for both.

The combination of complex interaction and on-demand computation of derived data at-

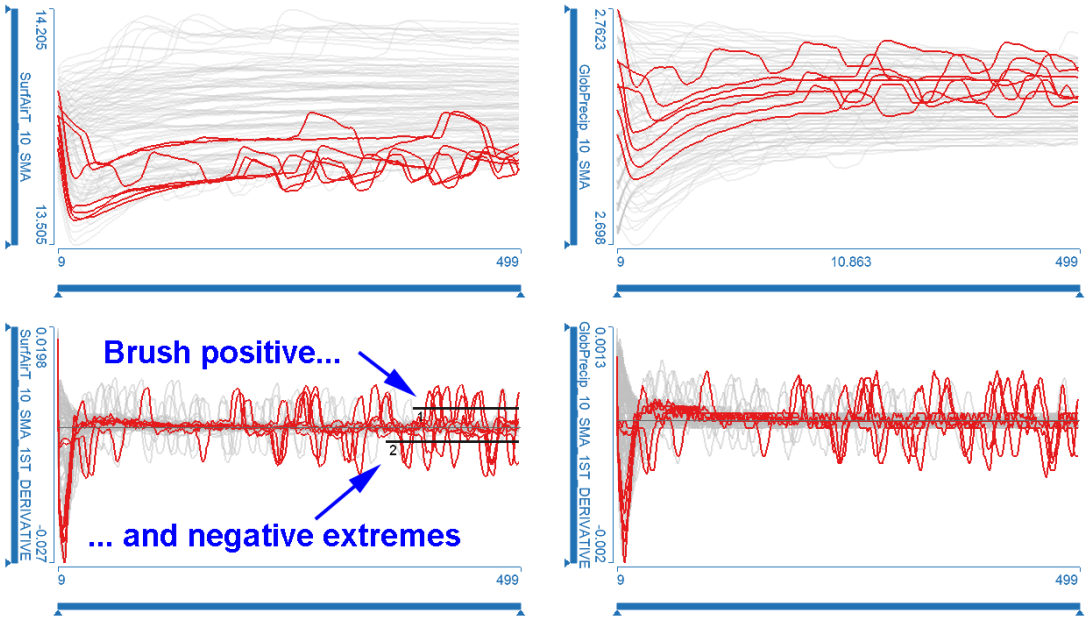


Figure 4.4: An example of a level three analysis. In a first step, the user triggered the computation of first derivatives of the temperature and precipitation curves. The plots show temperature and precipitation (top left and right) and their first derivatives (bottom left and right). Two line brushes select extreme temperature derivates near the end of the simulation run. The jagged temperature curves become visible, including two that were completely hidden in Figure 4.3. The highlighted precipitation curves are similarly jagged, indicating correlation.

tributes constitutes the *third level* of visual analysis, shown in the rightmost circle in Figure 4.1. In order to compute the derived data for the next iteration of the analysis, the analyst should not need to interrupt the analysis session and use some different tool to perform computations. Such detours can significantly hinder the analysis. Quite the contrary, the computation must be tightly integrated into the IVA framework. Therefore, while the data in levels one and two of the visual analysis remains static throughout the analysis session, it becomes dynamic in level three. Data change as a result of the on-demand computations—for each computation, a new dimension (a column in the table in Figure 3.2(b)) is added to the data.

Note that each of the three circles in Figure 4.1 encloses the previous ones from the lower level(s), indicating that a seamless transition between circles (levels of analysis) is possible. For example, after a new derived attribute is computed (third level), the analyst can open a new view and combine a selection on the new derived data with already existing selections (transition to the second level). A brush in one of the views can be moved and the highlighted selection can be observed in other views (transition to the first level).

It is worthwhile to discuss the expected response time of the visual analysis application during analysis at different levels of complexity. Card et al. [41] have identified three thresholds of response times in human-computer interaction. If the application responds within 0.1 seconds,

then the response is perceived instantaneous. If the response takes longer, but less than 1 second, then the delay is noticed, but the user's flow of thought remains uninterrupted. If the response takes longer than 10 seconds, then users find it difficult to focus their attention and their flow of thought is interrupted. Actually, these findings resonate with a much older study by Miller [174], indicating that the pace of human cognition has not changed significantly for decades.

An interesting analogy can be drawn between the three levels of complexity in analysis and the three time constraints in human-computer interaction. The users' actions in level one analysis are spontaneous; they move a brush and observe the changes. The cycle time of the analysis loop is very short and many iterations are done in a short time. The user expects instantaneous response from the computer, so that the exploration remains fluid. At level two, the users' actions are more complex. When they choose to open a new view, they need to think about the type of view and the data attribute. When brushes are combined, they need to consider possible logical operators to use. The pace of action becomes slower, and slower computer response can be acceptable. Nevertheless, if the system does not respond within one second, then the user's cognitive process is interrupted. At analysis level three, the user can trigger the computation of new data attributes when he or she realizes that those can advance the analysis. This decision often reflects elaborate thinking, involving a broader overview of the analysis process. Users will likely accept longer response times, provided they have some feedback of the computer's actions in the form of a progress bar or the visualization being progressively updated.

4.3 Analysis of Families of Function Graphs

In this section we illustrate the above described principles in the context of simulations in automotive industry. Current emission regulations and very high efficiency requirements for modern car engines lead to very complex engine designs. Engineers have to deal with many, often contradicting, parameter settings. The use of simulation is unavoidable in modern engine design. Advances in simulation and computation technology have made multiple simulation runs, or simulation ensembles [118], possible. The idea is to run simulations for the same simulation model with different sets of control parameters. Interactive visual analysis is a perfect method to analyze and explore the resulting data sets. In our previous work, we have analyzed two important systems, a Diesel fuel unit injector [161] and a variable valve actuation system, by means of interactive visual analysis. A complete description of the two case studies is out of the scope of this dissertation. We rather give a short summary to provide context for this section. Interested readers can find additional information in the referenced literature [82, 204].

Variable valve actuation (VVA) [204] is an active research field in automotive industry. The operation of four-stroke internal combustion engines consists of intake, compression, power, and exhaust strokes. During the intake stroke, the inlet valve opens and vaporized fuel mixture enters the combustion chamber. During the exhaust stroke, the exhaust valve opens and the exhaust gases leave the combustion chamber. Conventional systems use a camshaft, where cams open and close the valves at specific times, dependent on the mechanical construction of the cams. Different operating conditions (e.g., load or engine speed) require different valve opening and closing timings to achieve optimal efficiency and emission. In engines equipped with a VVA mechanism, the valves' opening and closing times, as well as the valve lift can be controlled

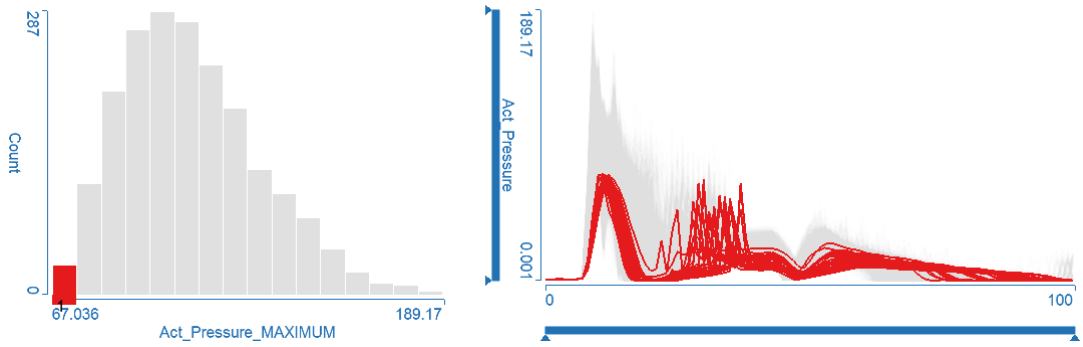


Figure 4.5: The histogram depicts the maximum of the pressure in the valve actuator. The items with the smallest maximum are brushed.

based on the load and engine speed. We explore the design of a hydraulically supported cam operated VVA system. Such a system has a camshaft with cams, and an additional hydraulic mechanism which opens the valve independently of the cams' position. Approximately 600 simulation runs were computed with different parameter settings and the ensemble data was analyzed.

Besides valve actuation, the *fuel injection subsystem* [82] is another engine component that has crucial impact on emissions and performances. Currently, the two most important types of injection systems for Diesel engines are the common rail and the unit injector systems. Common rail systems keep fuel pressurized to the injection pressure in a fuel rail which feeds the cylinders. The rail is common to all cylinders. In contrast, unit injector systems have the high pressure fuel pump integrated with the injector. There is one injector/pump unit per cylinder in the cylinder head assembly. We analyzed [161] the Delphi E3 Diesel Electronic Unit Injector (EUI) [82], an advanced unit injector with two independent, fast-response precision actuators that can change the injection pressure level and adjust the fuel delivery timing and duration, allowing a very flexible choice of fuel injection characteristics. 2880 different simulation runs were computed and analyzed in the case study [161].

In the following, we describe four of the most interesting analysis patterns detected in the two case studies. These patterns are applicable to virtually all domains where families of function graphs are analyzed.

4.3.1 Aggregates and Thresholds

Some of the typical goals in the analysis of families of function graphs can be effectively tackled by computing some aggregate of each member curve of the family. The most important aggregates from the IVA perspective are minimum, maximum, arithmetic mean, percentile (which also includes median), and integral. These aggregates are easily computed, yet they convey useful information. Aggregates are scalars that represent an entire function graph, significantly reducing data complexity. Thereby, standard visual analysis tools that are available for scalar data can be used in the exploration of families of function graphs. This possibly requires less

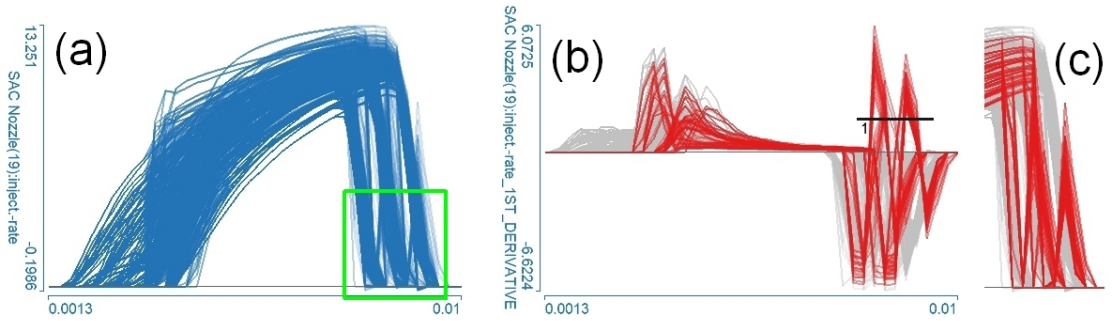


Figure 4.6: (a) Over 2800 overlapping function graphs depicting injected fuel rate. There are some curves rising near the end (region marked by the green rectangle), but they cannot be identified because of occlusion. (b) The user requested the computation of first derivatives. The slopes of interest can be brushed easily in the plot of the first derivatives. (c) A zoomed view of the selected curves with rising parts.

complex displays and simpler interaction, too.

The most often used aggregates in engineering are certainly minimum and maximum. It is interesting that analysts are often interested in function graphs with the smallest maximum or the largest minimum values. Such constraints describe curves that do not go over or fall below specific *thresholds*. The thresholds may represent desirable cases or outliers that need to be avoided. In Figure 4.5, the maximum values of the pressure in the valve actuator were computed. The reduced data complexity allows selecting function graphs with the smallest maximum in the histogram—a view much simpler than the ones normally used for families of function graphs.

4.3.2 Exploring Slopes

In many fields, a typical task when analyzing families of function graphs is finding curves that rise or fall at a certain point in time. This would not be easily possible with only level one and level two IVA. It would require stopping the analysis process, precomputing first derivatives, loading the data set again and then using another function graph view to brush positive or negative first derivatives at the time point. Level three, on demand attribute derivation or advanced interaction, makes this kind of query much easier and faster, without interrupting the IVA process.

During the analysis of the Delphi E3 EUI we want to achieve the high power mode of operation. The injection rate curve has to rise and decrease very steeply. Furthermore, the injection pressure must be as high as possible in order to inject the sufficient amount of fuel.

Before analyzing the control parameters causing the desired slope, we want to make sure there is no second needle opening, an undesirable phenomenon that happens sometimes. The needle is opened once more at the end of the cycle. This leads to an unwanted, uncontrolled subsequent injection, resulting in the rapid deterioration of the quality of the combustion. In the data this phenomenon appears as short rising sections in the injected fuel rate function graphs. There are more than 2800 overlapping curves in the function graph view in Figure 4.6(a), and we cannot see if there are any with rising sections near the end. In order to examine such cases

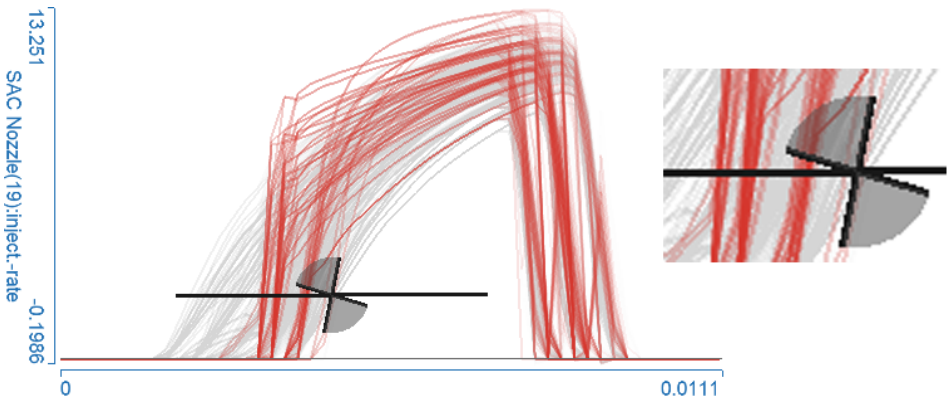


Figure 4.7: The angular line brush selects curves that intersect it at a given threshold of angles. Steep rising curves can be selected. The angular line brush is shown magnified on the right.

we use the first derivative of the injected fuel rate function graphs. The user simply "orders" the first derivative of the family of function graphs and a new dimension is created in the data set. A new function graph view is opened in the CMV where the first derivative curves are depicted (Figure 4.6(b)). We can see positive first derivative at the beginning of the injection cycle—rising curves, as expected. At the end of the injection cycle we can see curves with negative values (falling curves, as expected), and also positive ones. Positive derivatives here are unwanted; those are the curves we wanted to identify. We can easily select (and also exclude) them now using a simple line brush.

We can also select function graphs having a certain slope by enhancing the interaction. The *angular line brush* [161] is proposed as a method for the intuitive brushing of the function graphs based on the angle. As the function graph view already has its own type of brush, the line brush, which is proven and well accepted, it seemed as a best solution to improve it to be able to brush function graphs with a specific slope interval. In parallel coordinates, angular brushing [89] has been proposed as a method to brush lines that have particular slopes. In our case, the user can simply define a range of angles on the line brush. Only function graphs that cross the line at an angle within that range are selected. Figure 4.7 illustrates a case from the EUI analysis where steep rising curves are brushed. Although we can allow the angle constraint for arbitrarily oriented line brushes, it proved to be most intuitive in combination with either horizontal or vertical ones. For arbitrarily oriented line brushes, the user would need to mentally combine the angle constraint with the slope of the line brush in order to brush particular slopes. The angular line brush is used in the original function graph view, so it saves valuable screen space. The alternative approach, computing the first derivate and using a simple line brush requires an additional view.

Once the first derivative is computed it can be used as input to aggregation. When looking for function graphs which do not rise or fall significantly, the first derivative can be computed first, and then its minimum and maximum scalar aggregates. The scalar aggregates are depicted in a scatter plot and function graphs having high minimum and low maximum of the first derivative can be brushed now. These are flat curves. Figure 4.8 illustrates such a case.

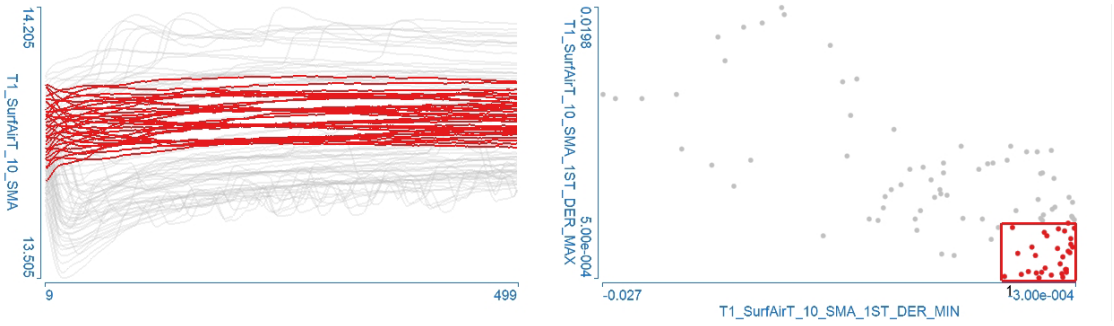


Figure 4.8: The analyst is interested in temperature curves that remain "flat", i.e., do not have steep rising or falling sections. The minimum and maximum values of the first derivatives are computed and depicted in the scatter plot. Low maximum and high minimum values of the first derivative can be brushed to select the curves of interest.

Brushing slopes is a premium example of the two identified approaches for improving IVA. We described both approaches, i.e., attribute derivation and using standard interaction methods; versus introducing new interaction methods that work with the original data. They are equally intuitive and both have their advantages, depending on the task.

4.3.3 Exploring Shapes

In many cases, engineers are looking for function graphs of certain shapes. Sometimes a combination of several line brushes is sufficient to isolate curves of desired shape [139], but most often a more advanced approach is needed. This problem, too, can be solved by using more advanced interaction, or by computing several specific aggregates and using simple brushes. The well-known similarity brush [35, 96, 178] represents the solution using advanced interaction. We propose two ways to perform similarity brushing: the user sketches the shape and then all similar curves are selected; or the user picks one of the curves and all curves similar to that one are selected. Various smoothing methods are available to alleviate the detrimental effect of noise on the performance of the shape recognition algorithm. The tolerance used in curve comparison can also be specified.

An example from the analysis of variable valve actuation simulation is shown in Figure 4.9. We are interested in a specific shape of the function graphs: quick rise, a certain span of time while the valve is opened, then quick fall to a given range, and finally smooth closing. In Figure 4.9(a), the shape is defined using a similarity brush composed of four segments. The similarity thresholds can be specified per segment. This facilitates very precise control over shape. As an alternative, we can compute four scalar aggregates, *r1*, *pw*, *r2*, *br*, shown in Figure 4.9(b). The aggregates are visualized in a parallel coordinates view in Figure 4.9(c) and we can brush desired values for all aggregates. A similar set of function graphs is selected again, shown in Figure 4.9(d). We needed specific aggregates in this case, and an additional view to define the shape via aggregates.

Less complex shapes are often easier brushed by several angular line brushes, or by using line brushes on the first derivative. Using the first derivate for brushing shapes requires an

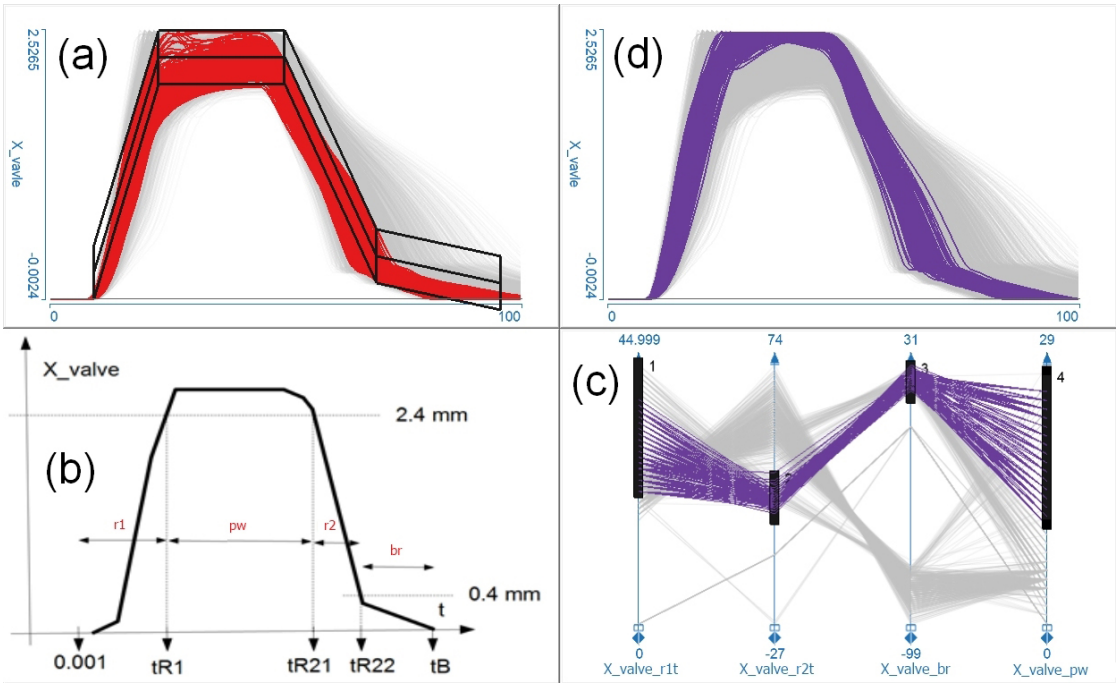


Figure 4.9: (a) A specific shape of function graphs is brushed by a similarity brush that consists of four segments. Similarity is defined in terms of maximum allowed deviation from the shape. (b) A set of scalar aggregates—timings of characteristic points—that describe shape. (c) Specific shapes of function graphs can be brushed by selecting ranges of the scalar aggregates in the parallel coordinates. (d) The same set of function graphs is highlighted as by the similarity brush.

additional view. Curvature, the amount by which a line deviates from being straight, is also proposed as a means of finding curves of certain shapes, although it is useful in only some special cases.

4.3.4 Cross-Family Correlations

Up to now, families of function graphs have been depicted in separate views. We have used the CMV system to compare multiple families; see also Section 3.4.3. That is not always sufficient. We use an example from the VVA simulation. The hydraulic model of the VVA must be developed based on the energy conservation of the complete system (hydraulic power unit composed of engine and hydraulic pump, the accumulator, and the valve system composed of the valve actuator and the valve itself). The complete hydraulic valve actuator model is simulated. From among the approximately 600 simulation runs, we are looking for the runs where energy consumption is low.

The data contains three families of function graphs: valve lift, actuator volume and actuator pressure. We can depict them using three function graph views. First we select the desired valve lift shape. We can see the corresponding shapes of the actuator pressure and volume curves. The

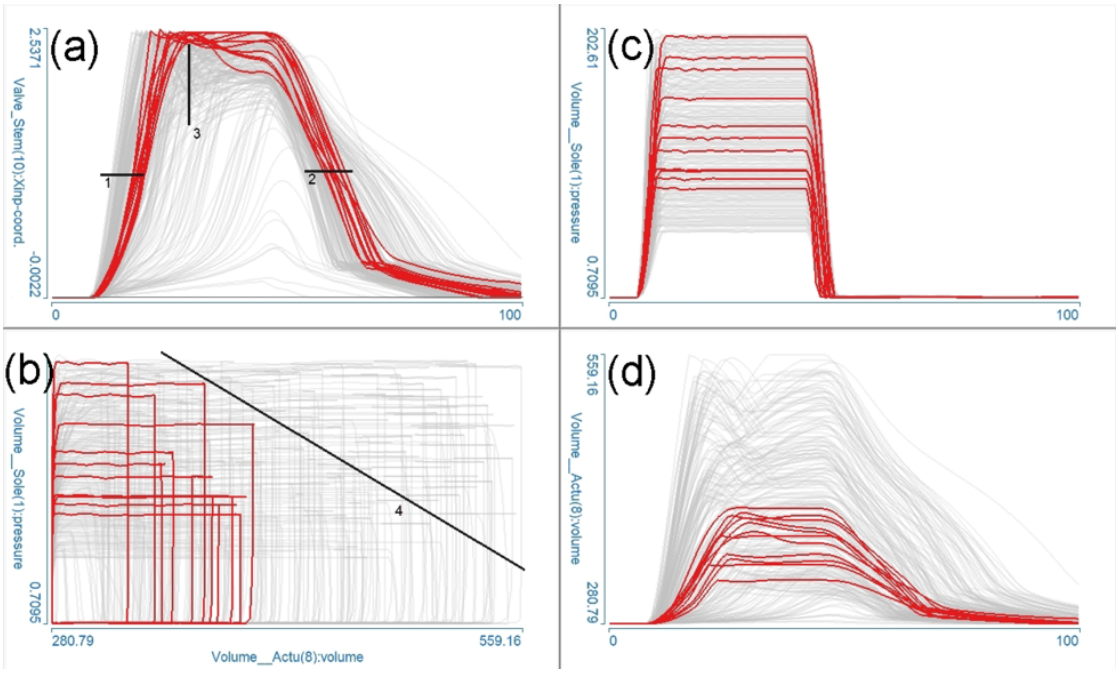


Figure 4.10: (a) The desired valve lift curves are selected by a combination of three line brushes. (b) The phase diagram simultaneously displays pressure and volume. A simple line brush excludes high energy consumption cases. (c) and (d): Corresponding pressure and volume curves.

energy consumption depends on both actuator pressure and volume.

We propose using phase diagrams—a plot often used in physics—to visualize two families of function graphs with a common independent variable (e.g., time) in a single view. The horizontal and vertical axes of the phase diagram represent the two families. Points are plotted by using the corresponding values of the function graphs in the two families as horizontal and vertical coordinates. A point is plotted for each value of the common independent variable. Successive points are connected by line segments. The result is one line showing values of the two function graphs. The process is repeated for all other pairs of function graphs from the two families. This plot often reveals interesting relations between different dimensions of the data set that would be more difficult to discern using only function graph views. The phase diagram is also fully interactive. Like the function graph view, it supports angular line brush and similarity brush which makes finding and brushing hidden relations possible.

In Figure 4.10, pressure and volume are depicted in the phase diagram. The area outlined by each closed curve in the phase diagram corresponds to the energy used in one simulation run. Once the function graphs with desired lift shapes are selected in a separate function graph view, we can refine the selection in the phase diagram by excluding cases with large energy consumption.

4.4 Chapter Conclusions

Data sets from many different problem domains contain families of function graphs, and interactive visual analysis represents a premium methodology for their analysis. We described three levels of complexity in IVA. The first level is represented by linking and brushing with one brush. On the second level, more views are used and brushes can be combined with logical operators. The third level calls for the seamless integration of attribute derivation and advanced interaction.

An advantage of using advanced interaction is that it does not increase the amount of data and the visual complexity, because usually no additional views are necessary. On the other hand, it generates an additional cognitive load, because the user needs to manage the advanced interaction method mentally. New types of analysis tasks may require new, specialized interaction techniques. Designing specialized advanced interaction mechanisms requires a priori knowledge of the expected analysis tasks, which is often not available.

In contrast, attribute derivation increases the amount of data by generating additional synthetic data attributes. The visual complexity is usually increased, because typically new views are necessary to display the derived data. It is essential that the analysts understand the meaning of the derived attributes. In fact, the derived attributes are often exactly the ones that are used in computational analysis, thus experienced analysts are familiar with them. Well-known, simple mechanisms can be used to interact with the data. Due to the step-by-step approach, with a sufficiently rich set of basis operations, a lot of very different derivations are possible, also ones that were not necessarily anticipated at the time when the IVA system is designed. This opens possibilities to solve unforeseen analysis tasks, too. The step-by-step, iterative procedure is more aligned with the iterative nature of interactive visual analysis. The series of visualizations generated when using attribute derivation can be used to discuss findings and communicate the analysis procedure. On the other hand, when advanced interaction patterns are employed, the analysis process is not intuitively captured in the visualizations but it needs to be documented in some other manner.

The advantages and drawbacks of the two essential building blocks, advanced interaction and attribute derivation, are complementary. There is no universally better choice. In open and flexible IVA systems that can be used for a variety of problems, attribute derivation may be preferred. However, in more targeted IVA solutions, where similar problems need to be solved repeatedly, advanced interaction can be more time-efficient for the experienced user.

Chapter 5

Additional Views for Families of Function Graphs

"Nothing's beautiful from every point of view."

— Horace (65–8 BC)[1]

In the previous chapters we have used the function graph view to display families of function graphs. In this chapter we introduce two novel views, the *segmented curve view* and the *color lines view* that can also display families of function graphs. They improve certain shortcomings of the function graph view. The *segmented curve view* can effectively display distributions in families of function graphs while also preserving outliers. Moreover, it avoids suggesting continuity in the data, which can be misleading when the independent variable is not continuous, e.g., it is frequency. For a fixed value of the independent variable of the functions, a bar extends from the minimum to the maximum values across the family of function graphs. Each bar is divided into segments (bins) that are color mapped to represent the number of function graphs with the value in that segment. We also propose the *color lines view* to better visualize patterns and clusters in families of function graphs. The color lines view provides a pixel-based, two dimensional, rectangular view where each row of pixels represents a single function graph. The horizontal point positions in the row correspond to values of the independent variable. The point colors represent the corresponding value of the function graph. The rows, placed directly above one another in parallel, show a family of function graphs. The color lines view offers sorting and brushing features which support visual analysis procedures that are difficult to perform with the function graph view.

5.1 Motivation

In the previous chapters we have used the alpha-blended *function graph view* to display a family of function graphs. This approach proved to be very effective. The function graph view was

[1] Quintus Horatius Flaccus, Roman lyric poet during the time of Augustus.

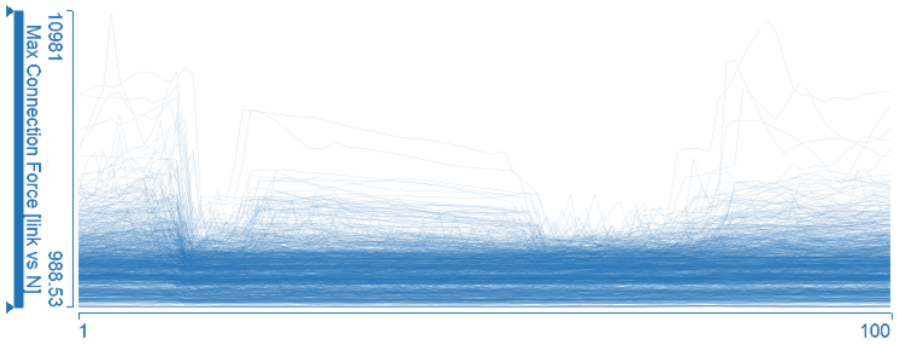


Figure 5.1: Function graph view of a family of function graphs. The x axis represents the chain link index (1–100). The y axis indicates the maximum connection force at the given link. In this function graph view there is one function graph for each of the 1,152 simulation variants, producing an overlay of 1,152 individual function graphs of maximum connection force.

used to successfully display, explore and analyze data sets containing more than 40,000 function graphs per dimension.

Unfortunately, the function graph view may fail to reflect certain properties of some data sets containing families of function graphs. The continuous lines suggest continuity in the displayed data series. If the actual data are continuous in nature, then connecting the sample points with continuous lines in a function graph plot is a valid and meaningful approach. However, if the data are not continuous, then the visualization is semantically incorrect. Choosing the right transparency factor in an alpha-blended function graph overlay is always a difficult task. A transparency setting that reveals the distribution in densely populated regions makes outliers barely visible. Conversely, a transparency setting that preserves outliers makes dense regions indiscernible because of overdraw. This problem is exemplified in Figure 5.1. We address these issues in Section 5.2 by proposing a novel view, the segmented curve view, for the visualization of families of function graphs.

When lines in the function graph view cross each other then it is often not easy to trace them. For example, assume that the values of several function graphs in a family are relatively close at a given t_1. In other words, there is a cluster at t_1. It is often difficult to assess if they are similarly clustered at a different time t_2, and if yes, then whether their relative ordering is the same at t_1 and t_2. Using the formalism introduced in Section 3.2, the question can be also expressed as: does $f(\mathbf{x_1}, t_1) < f(\mathbf{x_2}, t_1)$ imply that $f(\mathbf{x_1}, t_2) < f(\mathbf{x_2}, t_2)$? In Section 5.3 we propose another novel view, the color lines view, with interaction features to support such analysis tasks.

5.2 The Segmented Curve View

We propose the *segmented curve view* to overcome some of the shortcomings associated with the function graph view. A part of a timing chain simulation data set is used in this chapter as an illustrative example. The independent variable of the function graphs in this data set is not time, but the index of the chain link. Please refer to Section 7.2 for a more detailed description

of this data set. The maximum connection forces between neighboring chain links are shown in Figure 5.1. There are 100 links in the chain, thus the horizontal axis is labeled from 1 to 100. There are 1,152 different simulation variants with different settings of design parameters. Each simulation variant is represented by one function graph, thus the function graph view shows a family of 1,152 function graphs. We use the data set in Figure 5.1 to introduce the segmented curve view so that direct comparisons between the two views can be made.

It is often necessary to use aggregation methods to reveal characteristics of large data sets. An aggregation-based alternative to function graph displays is proposed by Andrienko and Andrienko [9]. Their approach displays the distribution very well, but fails to preserve the actual values of the function graphs. Johansson et al. [108] have proposed using different transfer functions to preserve and highlight outliers and clusters in parallel coordinates.

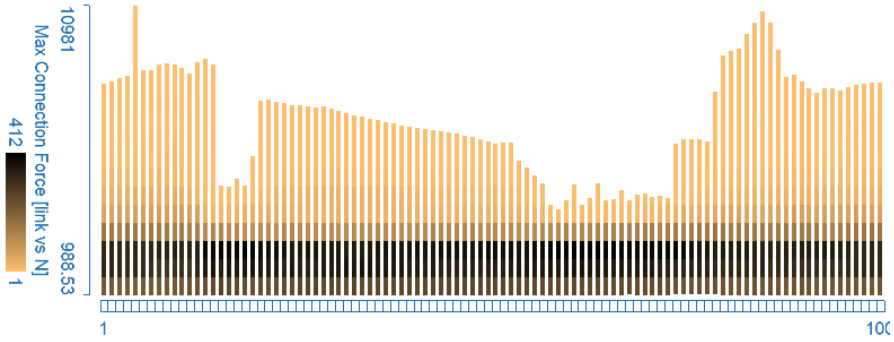
The process of binning [205] partitions the data space into intervals (bins) and assigns an occupancy value to each bin indicating the number of data items that belong to the bin. In other words, the "new data" represents a count of the original data items in each bin. A histogram is one example of binning. The width of each bin and the offset of where the bins start determine how the histogram appears. There are many other applications of binning, for example, the identification of outliers in parallel coordinates visualization of large data sets [185].

The segmented curve view is related to histograms and histogram-like frequency-based visualization techniques. For each fixed value of the independent variable, a bar extends from the minimum to the maximum values across a family of function graphs. Therefore, the bars depict minimum and maximum values of the dependent variable for each independent variable. The individual bars provide segmentation along the independent axis and avoid the possibly false suggestion of continuity. Similar approaches have been used, for instance, in the Information Mural [105]. However, we extended it and instead of showing only the minimum to maximum range, we also show the distribution of the dependent variables in a bar. Each bar is divided into segments (bins) where color represents the number of function graphs with the value in that segment. The new view provides a valuable insight into data containing families of function graphs, and offers, in some cases, significant advantages over the function graph view.

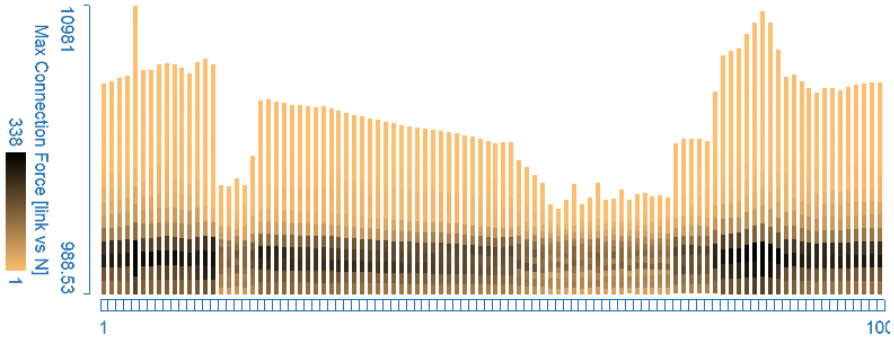
5.2.1 Segmentation and Binning

The segmented curve view provides two-fold segmentation of the displayed data. The individual bars for each value of t provide segmentation along the independent axis and avoid the possibly false suggestion of continuity.

In order to depict the distribution we introduce segmentation along the vertical axis. Each bar is divided into segments (bins) of uniform height where each segment is colored to indicate the number of function graphs passing through the segment. We use the same color mapping (the orange to black color gradient) throughout this section to provide a consistent presentation of views. Bins with the smallest curve count are shown in orange and the ones with the largest curve counts are black. Although a detailed discussion of color scales is not in the scope of this dissertation, we must mention that there are a lot of perceptual issues related to color mapping [150, 270]. The human eye distinguishes some hues and gradients better than others. The orange to black gradient was chosen because it seems to convey the density of function graphs well, and maintains reasonable contrast to the light gray used to indicate context.



(a) Global binning creates equal size bins in all bars, which makes direct comparison of bars easier, but fails to show fine details of distribution in shorter bars.



(b) Local binning creates the same number of bins in all bars, which reveals more information in densely populated areas.

Figure 5.2: (a) Global and (b) local binning in the segmented curve view using 16 bins. Orange bins contain few function graphs while black ones contain many. In (a) black bins contain 412 function graphs while in (b) they contain 338.

When segmenting a bar we can take the bar's actual minimum and maximum and uniformly partition this range. We term this *local binning*. Alternatively, we can partition the range defined by the overall minimum and maximum of all bars. We term this *global binning*. Figure 5.2 illustrates both binning strategies.

Global binning produces bins in different bars that have the same limits. That makes the direct comparison of distributions in different bars possible. However, when the minima and maxima of the bars are very different, then many bins will be out of that range and they remain empty. That leaves relatively few bins within the vertical bars and they cannot display the distribution with sufficient detail. We can create more bins to offset that and achieve a better resolution. The number of function graphs in the most populated bin will decrease and the color gradient will have fewer entries. Therefore, the visualization does not convey distribution very well.

One way to improve the visualization is to use local binning. Local binning creates the same number of bins in each bar. This provides a finer resolution for bars whose minimum

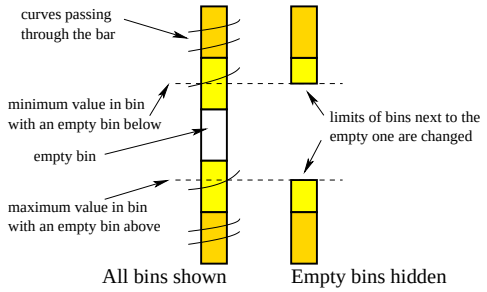


Figure 5.3: The process of removing empty bins. This figure shows one bar of the segmented curve view. The limits of the bins next to the removed one are set to the actual minimum and maximum values of the function graphs in them.

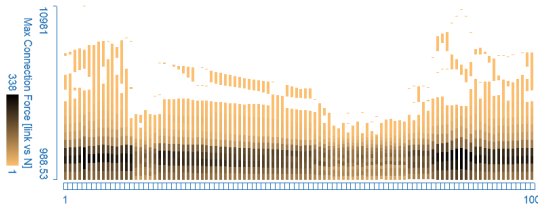


Figure 5.4: Empty bins are hidden using the idea shown in Figure 5.3. Compare to Figure 5.2(b).

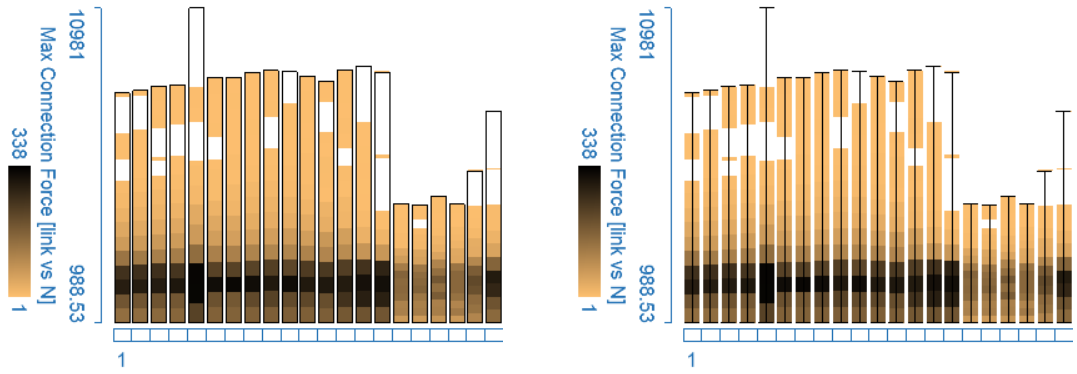
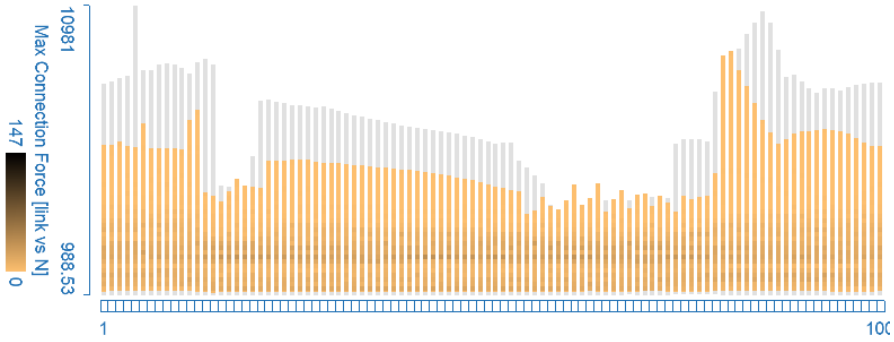


Figure 5.5: Framing the bars with rectangles (left) or boxplot-like lines (right) helps preserve the integrity of bars when empty bins are hidden.

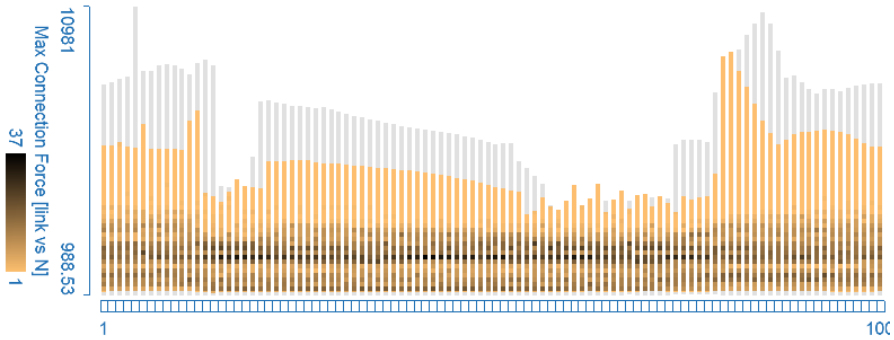
and maximum values are closer. Since the bin limits in different bars are different, comparing distributions in different bars is less straightforward. Consequently, global binning works better with more uniform minimum and maximum values, while local binning shows distribution in very non-uniform data using relatively few bins.

Some bins between the minimum and maximum of a bar may have no function graphs in them so they should not be visible. Bins beside empty ones are not drawn using original bins limits. If a non-empty bin has an empty bin as a neighbor, then the limits of the non-empty bin are moved to the actual minimum or maximum values of the bin. Figure 5.3 shows the procedure and Figure 5.4 shows the result.

We propose two possibilities to add frames to the individual bars in order to preserve their integrity when empty bins are hidden. A bounding rectangle (left in Figure 5.5) can be drawn around individual bars. If that makes the display too crowded, then we also offer framing lines (right in Figure 5.5), similar to box plots.



(a) Absolute color scale. The last color of the color gradient is mapped to the most populated bin in the entire family. The distribution is not visible very well.



(b) Relative color scale. The last color of the color gradient is mapped to the most populated bin in the current focus set. The distribution in the focus set becomes more visible.

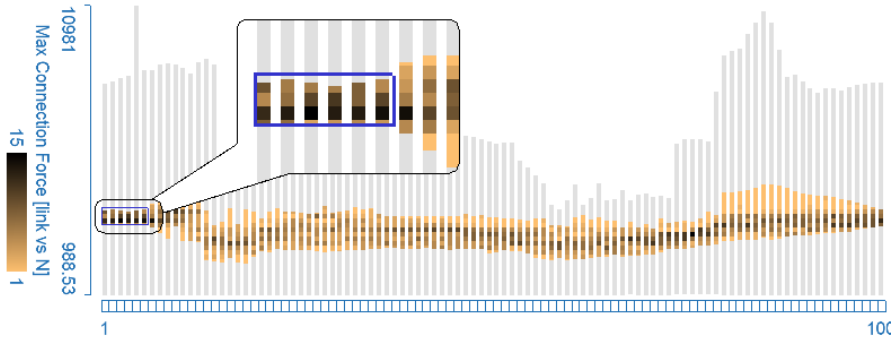
Figure 5.6: (a) Absolute and (b) relative color scales.

5.2.2 Color Mapping Strategies and Linking

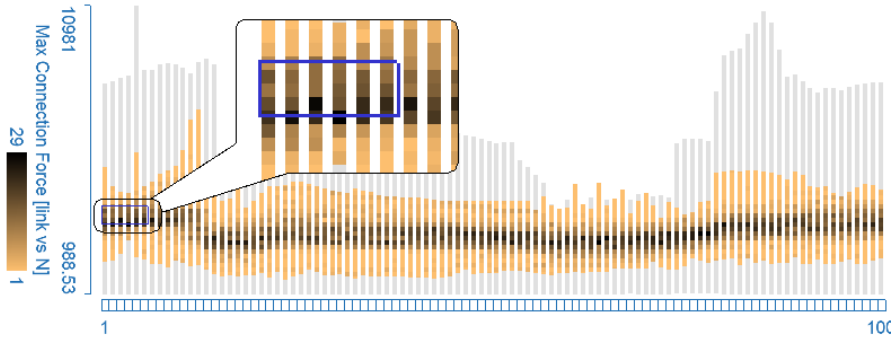
The natural approach to apply color mapping to the bins is to assign the first color of the color gradient to bins that contain one function graph. The most populated bin of all is drawn using the last color of the color gradient. This is termed *absolute color scale*.

The segmented curve view provides focus+context visualization. Figure 5.6 illustrates that the focus set is rendered using colored bins and the context is drawn in uniform gray. As described in the previous section, local binning is normally performed between the minimum and maximum values in each bar. In contrast, when a part of the data set is selected, then the local binning is performed on the range between the minimum and maximum values of the currently brushed items in each bar, thus more details of the distribution are displayed in the focus.

When the segmented curve view is linked to other views in a coordinated multiple views system, then the most populated bin in the focus set can contain considerably fewer curves than the absolute maximum. This means that only the first few colors of the color gradient are used for the displayed bins. That is a disadvantage of using the *absolute color scale*, shown in Figure 5.6(a). It renders the color gradient and thereby the distribution less visible. We can overcome this problem by mapping the last color in the gradient to the number of curves in the



(a) Function graphs that never leave the brush are selected by taking a logical AND of the brushed items in individual bars.



(b) Function graphs that enter the brush anywhere are selected by taking a logical OR of the brushed items in individual bars.

Figure 5.7: Comparison of the brushes spanning multiple bars. In both (a) and (b) middle ranges of the first six bars are brushed by the blue rectangle. The difference is how the sets of items selected in the individual bars are combined.

most populated bin in the actual focus set. This approach is called *relative color scale* and it is shown in Figure 5.6(b). It helps to reveal the details of the distribution in the focus.

However, using an absolute color scale has an important advantage: the same color represents the same number of curves regardless of how many items are actually brushed. This preserves the temporal coherence of the visualization also when the number of brushed items changes.

5.2.3 Brushing in the Segmented Curve View

We can brush a specific range in the segmented curve view to reveal finer details of the distribution. Local binning is performed on the brushed range to show more details and the brushed items are also highlighted in the linked views. We offer a rectangular brush that can span multiple bars, similar to the timebox widget [96]. There are two ways to interpret a brush that spans multiple bars.

- A function graph can be selected if it enters and leaves the rectangle through its vertical edges (Figure 5.7(a)). That is the intersection (logical AND) of the sets of function graphs selected in the individual bars.

- Alternatively, a function graph can be selected if it enters and leaves the rectangle through any edge (Figure 5.7(b)). This is the union (logical OR) of the function graphs selected in the individual bars.

5.2.4 Comparison with the Function Graph View

The function graph view suggests that the displayed data series is continuous while the segmented curve view does not. For example, the time scale of function graph data acquired via measurements or produced in simulation is inherently discrete. Depending on the semantics of the data, the segmented curve view may be a better choice for such discrete data. It is attractive in particular when connecting the sample points with continuous lines may be misleading. An obvious example is the Fourier spectrum of time series data.

Furthermore, the analysts are often interested in the minima, maxima and distribution of function graphs of a family. They want to see and compare the number of function graphs that fall within given ranges and discover patterns or outliers in the distribution. More specifically, they want to find out if there is any correlation between the distributions at different values of t. In Figure 5.7(a) for example, the relatively narrow and uniformly distributed range expands but still has a roughly uniform distribution around the middle of the horizontal axis. However, near the right end of the axis most function graphs are concentrated in the lower ranges.

One could argue that the segmented curve view is nothing more than a discretized version of the alpha-blended function graph view, especially if many bins are created with global binning. However, there is a very important advantage of the segmented curve view compared to alpha-blended function graph overlays: the colors assigned to the segments can be chosen from an arbitrary color scale.

This avoids a particular disadvantage of the alpha-blended function graph view. When many function graphs are displayed in alpha-blended function graph view, their transparency has to be increased to avoid clutter because of overdraw. However, it is often difficult to discern fine details of the distribution, especially in areas where only few function graphs pass through. The individual function graphs are simply too transparent to be visible. With the segmented curve view one can use a color scale for the segments that is clearly visible against the background and the visualization preserves both outliers and fine details in the distribution.

5.3 The Color Lines View

In this section we introduce another novel view for the visualization of a family of function graphs: the *color lines view*. Each function graph is displayed as a row of pixels. The horizontal positions in the row correspond to values of the independent variable of the function. The corresponding function value is indicated by the color of the pixel. Each function in the family is represented by one such row of pixels. The rows are placed directly above one another to display

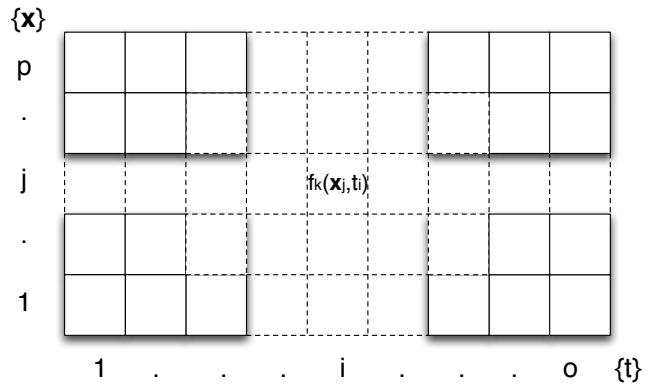


Figure 5.8: A rectangular arrangement of $o \times p$ pixels for a family of function graphs $f_k(\mathbf{x}, t)$.

a family of function graphs. The color lines view offers several sorting and brushing possibilities that make it a powerful technique for the visual exploration of clusters and patterns in a family of function graphs. This view is also integrated into a coordinated multiple views system, similar to the function graph view (Section 3.3) and the segmented curve view (Section 5.2).

The color lines view is a pixel-oriented visualization technique [122], similar to the Line Graph Explorer proposed by Kincaid and Lam [132]. Pixel-oriented displays simultaneously represent many data items by mapping each of them to the color of a pixel and arranging the pixels suitably. The value of each dimension is visualized as a colored pixel. The pixels belonging to each dimension are grouped into adjacent areas. Visualizations resulting from changes in the arrangement and the coloring of the pixels provide information on dependencies and patterns [122]. The proper arrangement of pixels is important to allow pattern discovery and support visual analysis. Data sets can have a natural ordering of elements (e.g., time-series data) or can be without ordering (e.g., categorical data) [122]. The perception of clusters and patterns can be further improved if the arrangement can also depend on the user's query [122, 238].

Mapping data values to color is a key aspect of pixel-based methods [122]. Color gradients can provide more just noticeable differences than grayscale ones. A single optimal color scale may not be feasible, but a combination of several interchangeable color scales can provide a better perception of the information [151].

5.3.1 Introducing the Color Lines View

The color lines view can display one function graph dimension of the data model introduced in Section 3.2. A function graph (data series over t) $f_k(\mathbf{x}, t)$ contains o data elements, one for each value of $t \in \{t_1, \ldots, t_o\}$. $\mathbf{x} = [x_1, \ldots, x_m] \in I$ represents a vector of the independent variables I. The number of function graphs in a family (p) is equal to the number of all possible values of $\mathbf{x}$, Therefore, a family of function graphs contains p function graphs, each with o data points, a total of $o \times p$ data points.

Figure 5.8 shows that these data points can be arranged in a rectangle. The color of the pixel at coordinates (i, j) corresponds to the value of $f_k(\mathbf{x}_j, t_i)$. Thus each function graph in the

family appears as a straight horizontal line whose color changes to indicate the function value. The color lines view shows the lines placed one above the other. This dense pixel representation provides a compact overview of a family of function graphs. If p or o is larger than the number of rows or columns in the display, then some aggregation method needs to be used in order to fit the data set into the available display space. The Information Mural [105] is one possible aggregation approach. When o is larger than the number of display columns, then neighboring points of a function graph are aggregated. Those points are generally similar in value, thus comparatively little information is lost due to the aggregation. Unfortunately, when p is larger than the number of rows, then the corresponding values of several different function graphs are aggregated into one pixel. Those values can be quite different, and therefore important details can be lost due to the aggregation. Ideally, at least the minimum and maximum values aggregated into one pixel should be indicated. Alternatively, the extrema-preserving downsampling described by Piringer et al. [196] can be adapted; or a distortion-based focus+context approach similar to the Table Lens [207] can be used. It should be noted that when p is only slightly larger than the number of rows, then it may be more practical to let the user simply scroll the view vertically instead of computing any aggregates.

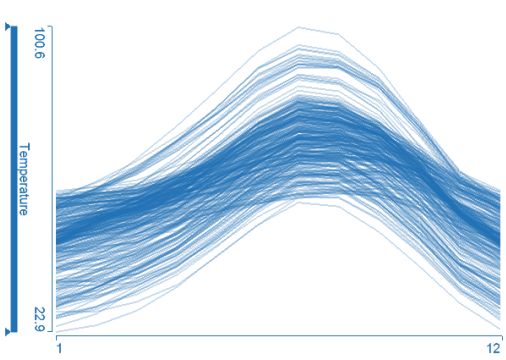
We use a simple meteorological data set to demonstrate this view. The data set consists of data from 303 weather stations in California[2]. Several attributes including station ID, longitude, latitude, and elevation are given for each station. Monthly average temperature and precipitation are given as functions over time. Note that this is a reduced data set and contains only 303 stations and covers only one year. It is used only for illustration purposes.

This data set matches the data model in Section 3.2. There is one independent variable, the station ID ($m = 1$) with 303 different values ($p = 303$). The dependent variables include longitude, latitude, elevation, temperature and precipitation ($n = 5$). Longitude, latitude, and elevation are scalars while temperature and precipitation values are data series of 12 points, one sample per month ($t \in \{1, \ldots, 12\}$, $o = 12$.

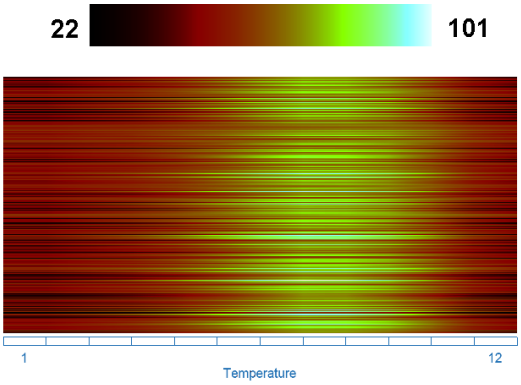
Therefore, there is one temperature function graph for each station. The set of all 303 temperature function graphs constitutes a family of function graphs. Figure 5.9 shows the temperature graphs in two different views, the function graph view and the segmented curve view. The blue to red color gradient is often used to represent temperature, but this color scale is perhaps not the best choice with respect to perceptual issues [30]. We use the *linearized optimal color scale* described by Levkowitz [150] throughout this section. Black indicates low temperature, then the scale goes through red and green, and finally cyan indicates the highest temperatures. The overall trend, cold winters and warm summers can clearly be seen in both views. In fact, the traditional function graph view is a clear winner for such an overview.

If we need a deeper insight into the data, then the simple overview provided in Figure 5.9(b) is not sufficient. However, if we can discover patterns in the data that are less obvious in other visualizations, then additional knowledge can be gained. The arrangement of lines obviously influences the patterns that appear [122]. If we provide an intuitive way of reordering them, then we can support visual analysis [238].

[2]World Data Center for Meteorology, `http://www.ncdc.noaa.gov/oa/wmo/wdcamet.html`



(a) Temperature data in a traditional function graph plot. There are 303 function graphs over 12 time points in the figure.



(b) The same data set in the color lines view. There are 303 lines and 12 segments in each line, following the arrangement in Figure 5.8. The color indicates temperature. The scale goes from black (22 °F) through red and green to cyan (101 °F).

Figure 5.9: Temperature data from 303 stations over a period of 12 months as a family of function graphs in (a) the function graph view and (b) in the color lines view. The horizontal axis represents the months January through December in both views.

5.3.2 Interaction with the Color Lines View

The visual analysis procedures, or search for patterns, involve transformations of the dense pixel display in a way that allows grouping of "similar" pixels while preserving the overall rectangular shape [23]. We assume that there is a natural ordering of the values of t so the transformations should preserve that ordering and keep individual data series intact. In other words, each row is an invariant of the transformations. Therefore, transformations are limited to swapping rows within the rectangle.

Given the initial ordering of rows, $1, \ldots, p$, the end result of a transformation is a permutation of those rows. The number of all possible permutations of p elements is $p!$, which is prohibitively large for exhaustive exploration. Therefore, some guiding criteria are needed for selecting a permutation. We suggest three criteria: internal sorting, external sorting (by a scalar dimension or by another function graph dimension), and brushing criteria.

Internal sorting

The internal sorting means that for a fixed value of t, t_i, we sort rows based on the values of their i-th pixels. The corresponding permutation π that is the result of the sorting of the values at the i-th pixel has the property:

$$f_k(\mathbf{x}_{\pi_1}, t_i) \leq \ldots \leq f_k(\mathbf{x}_{\pi_l}, t_i) \leq \ldots \leq f_k(\mathbf{x}_{\pi_p}, t_i)$$

where π_l is the row index with the l-th smallest value of f_k for the fixed t_i.

External sorting

External sorting takes another dimension, scalar or function graph, as guiding criterion. The external sorting by another function graph dimension does not use f_k but another function, $f_l(\mathbf{x}, t_i)$ to determine the order of the rows such that

$$f_l(\mathbf{x}_{\pi_1}, t_i) \leq \ldots \leq f_l(\mathbf{x}_{\pi_l}, t_i) \leq \ldots \leq f_l(\mathbf{x}_{\pi_p}, t_i)$$

The external sorting can also use a scalar dimension s as sorting criterion. s is either one of the independent variables $\{x_1, \ldots, x_m\}$, or one of the dependent variables $\{r_1, \ldots, r_{n_r}\}$. The rows are then sorted so that

$$s_{\pi_1} \leq \ldots \leq s_{\pi_l} \leq \ldots \leq s_{\pi_p}$$

Brushing

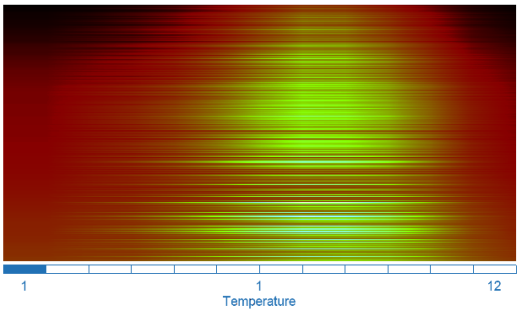
Brushing does not permute the rows, but marks a range of rows to be observed in subsequent permutations. The brushed rows are displayed using the default color scale. A different (grayscale) color gradient is used in the other rows. Therefore, the color lines view provides focus+context visualization where focus and context are shown in different color schemes. Furthermore, the view is split (Figure 5.11). The brushed lines are drawn separately under the view so that patterns in brushed lines can be explored in more detail. The brushed lines are also drawn in their original position to preserve their context information. This becomes relevant when the user brushes some rows, and then changes the sorting to use a different criterion. The positions of the brushed lines can potentially change because of the different sorting, but they retain their colors against the grayscale context. The arrangement of the brushed lines with respect to their context can convey valuable information, since it offers a means of comparing the results of the two different sorting criteria. This is exemplified in Figure 5.11(b).

5.3.3 Visual Analysis with the Color Lines View

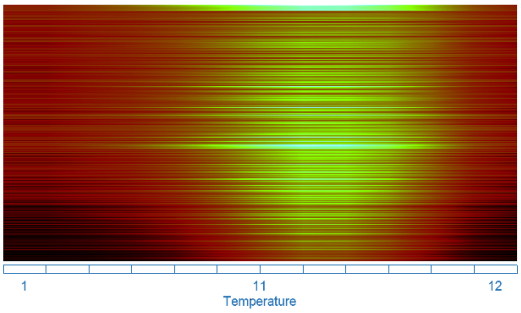
Visual analysis with the color lines view uses the interaction building blocks introduced in the previous section. Typical visual analysis procedures include:

- Sort rows based on the i-th pixel values and observe patterns at other pixel positions.
- Sort rows based on another variable and observe color patterns.
- Sort rows, brush a range of the sorted rows, perform another sort, and observe the distribution and color patterns of the brushed rows.

These simple procedures allow one to discover patterns in the shapes of the function graphs, correlations between function values at different values of t, and correlations between a family of function graphs and other dimensions of the data set. The focus+context visualization also effectively displays clusters with respect to the context.

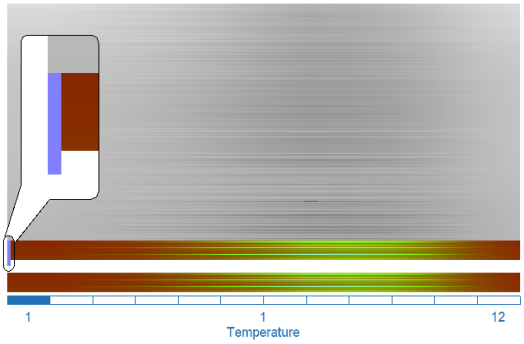


(a) The color lines view is sorted by January temperatures. This is indicated by the blue rectangle in the bottom left corner. The colors are remarkably varied in the middle, which means there is little direct correlation between January and summer temperatures.

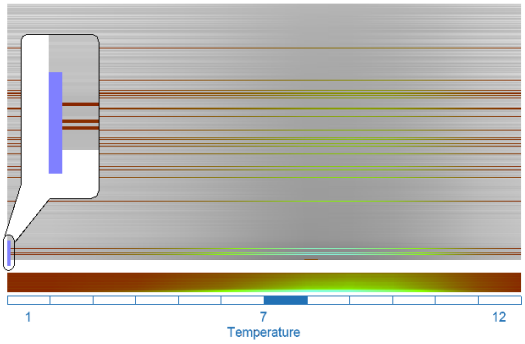


(b) The color lines view is sorted by elevation. Cold winters correspond to high elevations, but hot summers do not correspond to low elevation.

Figure 5.10: Sorting the lines by January temperatures (internal sorting) and by elevation (external sorting). The data and the color scale are the same in Figure 5.9(b).



(a) High January temperatures are brushed by the blue vertical line near the left edge of the view. The brushed lines are shown in the split view under the main one, too.

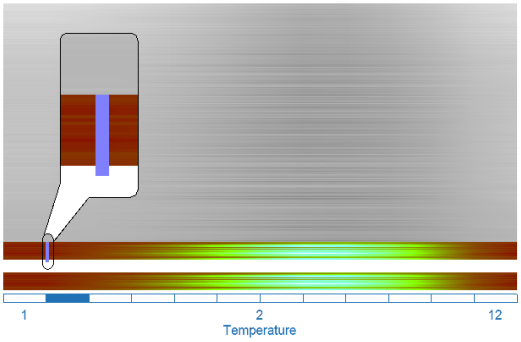


(b) The lines are sorted by temperature in July. The brushed set is quite scattered in the context, but the split view still allows its detailed exploration. The brush (blue line) persists to indicate that the highlighted items were clustered in January.

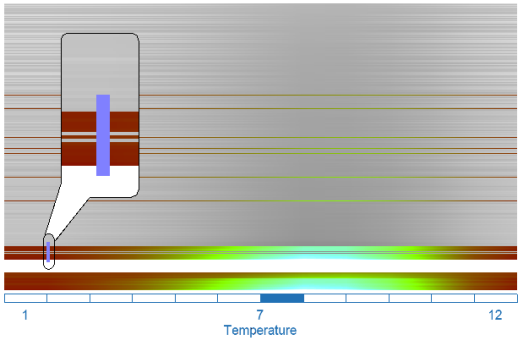
Figure 5.11: Brushing and sorting in the color lines view.

Finding patterns in function graphs

The basic interaction with the view is sorting the rows. The user can simply click a location on the horizontal axis representing the independent variable of the function graphs—in our example, months from January to December. Rows will be sorted and drawn according to the function values at this position. This is demonstrated in Figure 5.10(a). The color pattern in the clicked



(a) The lines are sorted by temperatures in February and high temperatures in February are brushed by the blue vertical line.



(b) The lines are sorted by temperatures in July. Most of the brushed set stays clustered together. The brush remains at February to indicate the sorting at the time of brushing.

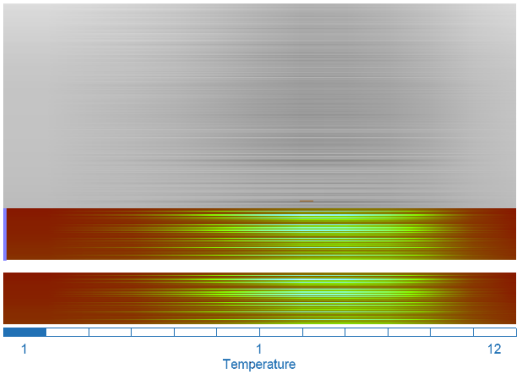
Figure 5.12: Brushing and sorting based on the February data. Compare this to Figure 5.11.

column is a monotonous transition. The color patterns in other columns may be similar, indicating correlations between the corresponding function values, or they may be quite different. Figure 5.10(a) shows that if the user selects January, the lines with low January temperatures are drawn at the top and the lines with high January temperatures are drawn at the bottom. We can see similar color patterns in January and in December, indicating that those temperatures are correlated, as expected. There are many lines at different rows that are red in the middle, which means low summer temperatures. Therefore, we can see that stations which are relatively warm in January are not necessarily warm in the summer.

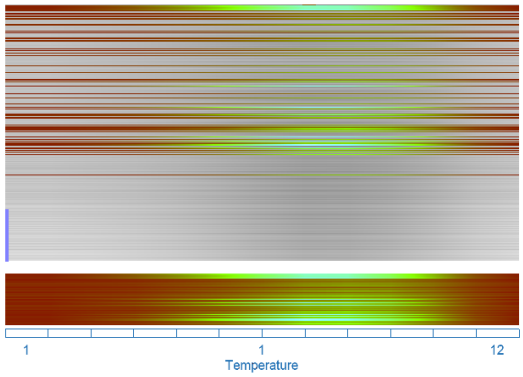
We can also brush lines in the color lines view to mark them as features of interest. Lines in the color lines view can be brushed by simply drawing a line across them. Remember that each line represents one item in the data set. Therefore, whole lines are selected and the set of selected lines is independent from the actual horizontal location of the brush. Actually, the brush line is displayed as a vertical line at a horizontal coordinate corresponding to the sorting position. For example, if the view is sorted by January and a brush is created at any horizontal position then it will be drawn in the left of the view. This is useful, because the sorting can be changed later, but the position of the brush indicates the sorting applied at the moment of brushing.

The highest January temperatures are brushed in Figure 5.11(a) and then the view is sorted by July (Figure 5.11(b)). The lines in the focus set are sorted in the lower part. We can also see those lines in the context where they are spread almost over the entire view. This means that mild January does not necessarily mean warm July in the data set.

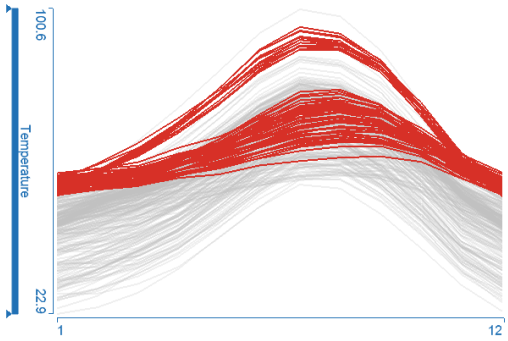
Figure 5.12 shows the same procedure for February. We sort the lines by values in February, brush warm temperatures and sort the lines according to July. Figure 5.12(a) shows the view with the highest February temperature selected. We sort the view for July (Figure 5.12(b)). The color lines view shows a peak for the July values. Interestingly, mild February does mean warm July at most stations!



(a) The color lines view is sorted by January temperatures (blue rectangle on the left) and high temperatures are brushed (blue vertical line on the left).



(b) The brushed lines sorted by elevation. Highest temperatures in January correspond to low elevation.



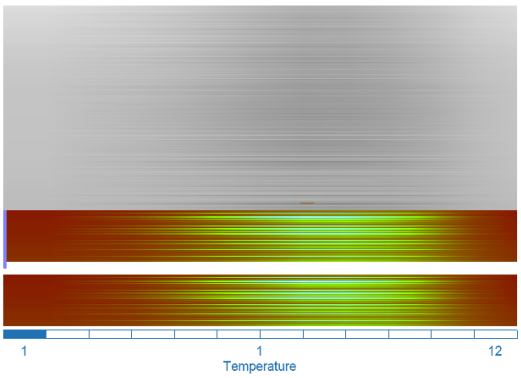
(c) The corresponding function graph view shows how curves with high temperature values in January (and December) diverge in temperatures for summer months.

Figure 5.13: Brushing and external sorting based on the January data and elevation.

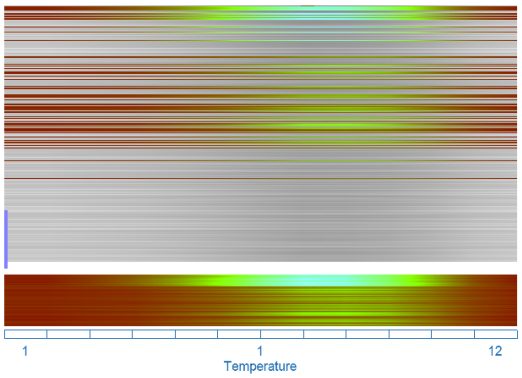
Finding inter-dimensional correlations

We can explore inter-dimensional correlations if we can sort the color lines view by some other dimension. Let us examine what happens if we sort temperature lines according to the elevations of the stations. High elevations are drawn in the bottom and low elevations in the top in Figure 5.10(b). We can see that cold winters (black at left and right) correspond to high elevations, as expected. On the contrary, hot summers (cyan in the middle) can be seen at many different elevations. The highest peak temperature is actually near the middle row. The correlation between elevation and high summer temperatures is less direct.

The correlation can be made more visible by brushing. The highest January temperatures are selected in Figure 5.13(a). Then the view is sorted by elevation in Figure 5.13(b). Low elevation is near the top of this image, high elevation is near the bottom. The majority of the brushed lines



(a) The lines are sorted by temperature in January and mild January temperature is brushed.



(b) The view is sorted by precipitation in January. Most of the brushed lines are at the top of the image, which indicates low precipitation.

Figure 5.14: Using external sorting by another function to reveal correlations between temperature and precipitation.

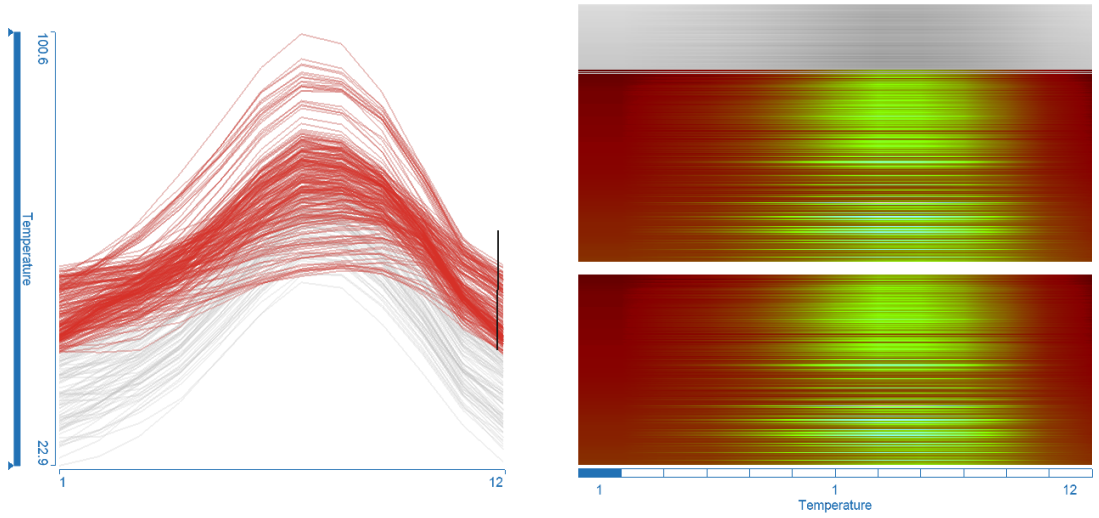
is near the top, thus we can see that high temperatures in January correspond to low elevation.

We can also find correlations between two function graph dimensions of the data set with the color lines view. We can use brushing and external sorting by another function graph. As an illustration, we try to find out if there is correlation between temperature and precipitation In January. The procedure is captured in Figure 5.14. The view is first sorted by January temperatures (Figure 5.14(a)). The lines with mild January are at the bottom. The mildest ones are brushed. The view is then sorted by precipitation in January. Data from stations where January is dry appears at the top and wet appears at the bottom. The brushed lines have moved to the top of the view in Figure 5.14(b). This means that the places where January is mild get little precipitation in January.

5.3.4 Comparison with the Function Graph View

One of the advantages of the color lines view compared to the function graph view is the better control over the number of items selected by a brush. In the color lines view, we can see precisely what percentage of the data set is brushed. Figure 5.13(a) shows a brush in the color lines view that selects 20% of the temperature values. Its length is exactly 20% of the height of the rectangle. Due to the overlapping of curves, it is not possible to determine the exact percentage of curves brushed in the corresponding function graph view (Figure 5.13(c)). The color lines view is more suitable for brushing a specific percentage of the total number of curves.

Figure 5.15 illustrates a different point. After selecting the top half of the temperature values in the function graph view in Figure 5.15(a), we discover in the linked color lines view (Figure 5.15(b)) that significantly more than one half of the curves are brushed. Therefore, the function graph view is better for brushing a given range of y-values.



(a) Selecting the curves above the mean value in December using the function graph view (black line brush on the right).

(b) The corresponding color lines view shows that significantly more than half of curves are selected.

Figure 5.15: The function graph view is more suitable for the brushing of curves within a range of values.

5.4 Chapter Conclusions

We introduced two novel views that address some specific problems and analysis tasks in data sets containing families of function graphs. Both views are integrated into the coordinated multiple views framework and provide focus+context visualization. The *segmented curve view* supports the analysis of distributions in families of function graphs, which is a common task. It can visualize fine details in the distribution while also preserving outlier graphs. Two different binning methods are proposed to support different goals. The global binning allows direct comparisons of the distribution at different values of the independent variable. Local binning, on the other hand, reveals more information in densely populated regions. Two color mapping strategies are available. The absolute color scale makes the visualization consistent when the focus changes. The relative color scale improves the resolution of the focus. Exploring patterns and clusters in families of function graphs is another common task in many problem domains. We have also introduced the *color lines view* that can display patterns and clusters in families of function graphs that are difficult to show with the function graph view. The sorting and brushing features provide support for complex visual analysis procedures, including the comparison of patterns and clusters in a family of function graphs at different values of the independent variable. The combination of brushing and sorting also allows finding relations between a family of function graphs and a scalar dimension, as well as the comparison of clusters between two families of function graphs.

Chapter 6

Interactive 3D Visualization of Multibody Dynamics

"Time and space are modes by which we think and not conditions in which we live."

— Albert Einstein (1879–1955)[1]

The focus of the previous chapters of this thesis has been on the exploration and analysis of data sets that include scalar dimensions and families of function graphs. Data has been plotted in a coordinated multiple views system in rather abstract views, such as scatter plots, parallel coordinates, and views displaying function graphs. Those abstract representations can be useful for many analysis tasks, especially if the data has no relevant spatial context. The analysis of a fuel injection system presented in Section 7.1 is one such example. However, if the spatial context of the data is relevant, then the analyst can often benefit from spatio-temporal visualizations. There is a lot of work published on interactive 3D visual analysis in the context of automotive engineering. Much of the literature focuses on the analysis of computational fluid dynamics [6, 37, 62, 146] or particle simulation data [281]. In this chapter we address a slightly different problem: the 3D visualization and visual analysis of rigid and elastic multibody systems. Such systems are commonly used to model and simulate certain components and aspects of internal combustion engines, but their visual analysis has perhaps received less interest to date. We describe an interactive and intuitive 3D visualization framework for the detailed analysis of rigid and elastic multibody simulation data. The proposed visualizations are iconic (glyph-based). We introduce glyphs representing vector attributes, such as force and velocity, as well as angular attributes including angular velocity and torque.

[1] German born American theoretical physicist who developed the general theory of relativity. Received the 1921 Nobel Prize in Physics.

6.1 Motivation

Simulation of rigid and elastic body dynamics has been a field of active research for decades [16, 18, 175]. The motion of certain parts of an engine (timing chain drives, for instance) can be simulated in software based on the concept of rigid body dynamics [241]. The vibration of the engine block and the stresses of the crankshaft can be computed using elastic body dynamics [154]. There is a relatively large body of literature on the simulation. However, the visual analysis of those simulation results has received less attention so far.

The concepts introduced in the previous chapters can be useful for the analysis of such simulation results. As a demonstration, a case study of a timing chain drive is presented in Section 7.2. An experienced engineer can certainly gather a lot of information from abstract views of the simulation results, and even from traditional, non-interactive, static 2D charts. Nevertheless, it is difficult and often even impossible to imagine complex 3D geometry only from such abstract representations. Spatio-temporal visualizations can improve the understanding of the simulated system's behavior significantly.

For instance, consider a sprocket and a chain link. The shapes of both objects are defined by the engineer and the simulation computes their positions and orientations. Without a physical view of the bodies, it can be very difficult to tell whether the chain link actually touches the sprocket, and if it does, then exactly where the contact point is. Similarly, rather all 3D vector quantities, like forces acting between sprockets and chain links, accelerations, etc. are difficult to imagine without physical views.

The motion of the bodies is described by the time-dependent simulation results. Visualization techniques for time-dependent data can be classified into two families: static (not time dependent) and dynamic (time dependent) visualizations [179]. Static representations, for instance, a plot of position as a function of simulation time, can allow the conclusion of quantitative statements. An example of dynamic visualizations is the actual animation of the bodies in 3D to show their motion. This puts the motion into a spatial context and provides an intuitive and qualitative overview of the system's dynamic behavior. 3D visualizations can also be useful when one wants to define a spatial focus of interest. If one wants to explore a certain part of the system, for example, the place where the chain leaves the sprocket, then he or she can simply zoom in there and explore simulation results of chain links passing by.

We collaborated with AVL-List GmbH (`www.avl.com`), a company developing industry-leading software for dynamic simulation of cranktrains. We have developed an interactive 3D visualization system where simulation results of multibody systems can be analyzed. The 3D visualization can convey important spatial information that is difficult to infer from more abstract representations. The 3D visualization has not been integrated into the coordinated multiple views framework introduced in Section 3; the reason in part being historic. Chronologically, the work presented in this chapter *predates* the rest of the thesis. Before the introduction of the 3D visual analysis system, the means of analysis for multibody simulations at AVL had been restricted to semi-automatically generated batches of static charts and diagrams. The interactive analysis and comparison of many simulation runs of multibody systems was not widely practiced, because available computational resources at the time were insufficient to compute more than a few variants in reasonable time.

6.2 Interactive 3D Visual Analysis of Rigid Body Dynamics

In this section we present a glyph-based system for the 3D interactive visual analysis of rigid body systems, in particular, for chain and belt drives. The simulation models of chain and belt drives in AVL's software are identical and the same attributes are computed. Therefore, it is justifiable to apply the same visualization methods to both drive types. In the following, we provide a very brief introduction to the use of rigid body based simulation in the context of chain and belt drives. Then we describe a glyph based visualization system that represents rigid bodies as well as scalar, vector and angular simulation result quantities. Finally, we illustrate how the system can be used in the context of timing chain drive analysis.

6.2.1 Rigid Body Simulation

Simulation of rigid body dynamics is an established model of real world dynamics that allows reasonably accurate simulation of certain systems at an affordable computational price. It has gained wide-spread acceptance in many fields of science, including virtual reality [224], robotics, and vehicle and spacecraft design [212]. The aim of some of those applications is producing convincing, "realistic looking", interactive animations, preferably in real time. On the contrary, others, including automotive engineering design applications, aim to maximize simulation accuracy and generally forsake real-time performance. In rigid body simulation [16, 175], real world objects are modeled by physically ideal rigid bodies that can have 6 degrees of freedom (translation and rotation along all three major axes), but cannot be deformed. The bodies are connected by connection elements that are basically springs with dampers. Bodies can also come into contact occasionally by colliding with each other.

Rigid body simulation is extensively used in the automotive industry [97, 225, 241]. We have cooperated with mechanical engineers at AVL-List GmbH and used AVL's *EXCITE Timing Drive* software to compute rigid body simulation data. This application is used in the design of timing drives in engines. Both chain and belt drives are handled in the same way in simulation. In chain drives, each chain link is a rigid body. The bodies are interconnected via stiff springs that model the pins. Belt drives are simulated by dividing the belt into many interconnected rigid sections. In the case of toothed belt drives, the sections generally contain one tooth. A typical chain drive contains 100-200 chain links. The simulation computes data for approximately 5000 time steps. For each time step and rigid body, e.g., chain link a, set of approximately 20 attributes is computed. The attributes include:

- displacement and orientation of bodies
- translational and angular velocity of bodies
- translational and angular acceleration of bodies
- forces and torques in connections between bodies
- relative displacement and relative velocity between endpoints of the connections
- locations of contact points between bodies, if they are in contact

Before the introduction of the visual analysis framework described in this chapter, the simulation results had been analyzed using static diagrams with little to no association to the actual

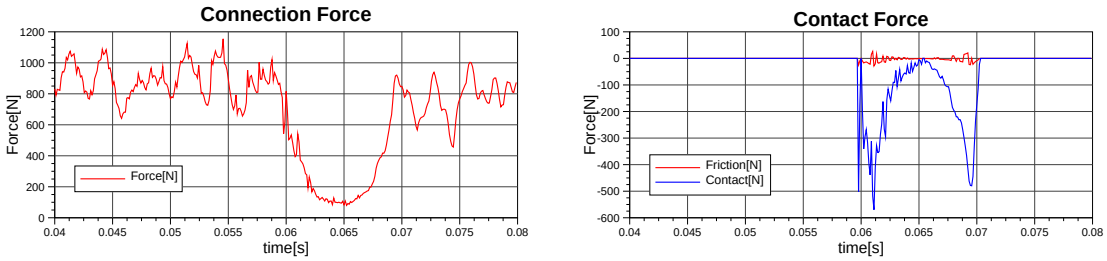


Figure 6.1: 2D charts showing (a) forces between two neighboring chain links and (b) forces between a chain link and sprocket.

3D models. Figure 6.1 shows two such diagrams of connection forces between two neighboring chain links and contact forces between a chain link and the sprockets. Note that these two charts represent one link only, and that there are about 200 such charts describing only one attribute, e.g., force between chain links, for a single simulation run. It is obvious that the number of diagrams produced for even a relatively simple system is huge and not easy to manage. Our goal is to develop a more intuitive 3D visualization system for rigid body simulation data which enables engineers to explore the data in ways not possible with 2D charting.

6.2.2 Glyph-Based Visualization of Rigid Body Dynamics

AVL's simulation model of rigid body systems does not use actual 3D models to represent the bodies. In fact, such geometry is often not available at all at that stage of the design process. The shapes of bodies are described by using a simplified set of *contours*. The contours represent the shapes of bodies that are relevant for the simulation. The rest of the shape is irrelevant for the simulation and remains undefined. This concept actually accelerates the modeling process, but makes the 3D visualization less straightforward, since there are no meshes to show. We propose glyphs to represent the bodies and connections; as well as the simulation results, including vector attributes (force, velocity, acceleration) and angular attributes.

Traditionally, glyph based visualizations often display more than one data attribute in one glyph [48, 54, 268]. Different visual properties of the glyph can encode different data attributes. For instance, the size and the color of an arrow glyph could encode two different attributes. Still we propose relatively simple glyphs that are used to visualize one attribute only. Several instances of the same type of glyph (preferably different variants, see below) can be created to visualize different attributes. We found that this offers more flexibility in the context of the chain and belt drive analysis.

It is well known that the placement of glyphs can also effectively convey information and improve or enhance the visualization [267]. We have chosen a data driven placement approach. Glyphs representing the motion of the bodies (velocity, acceleration) are always positioned at the center of gravity of the respective body, which we found intuitive. Glyphs representing data pertaining to connection and contacts between the bodies (force, moment) can be placed at the contact point of the two bodies or at the link point (see below) of either involved body.

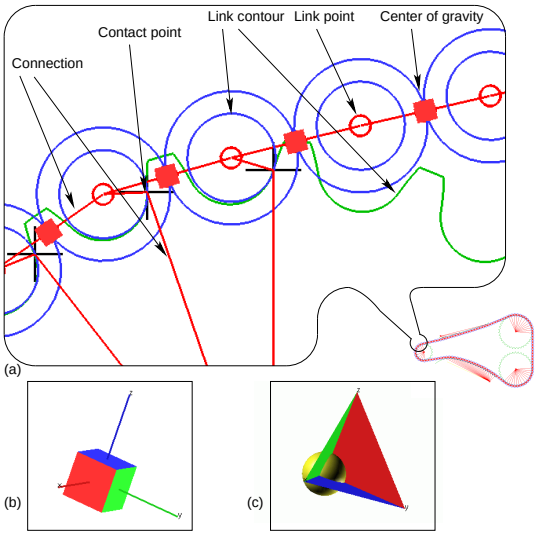


Figure 6.2: (a) Model of a chain drive. Overview and zoom of a part showing elements. (b) and (c) options for drawing center of gravity and local coordinate system.

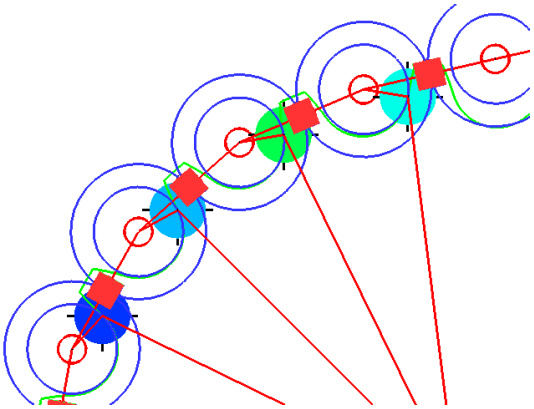


Figure 6.3: Color mapped disks visualizing the magnitude of contact forces between a sprocket and chain links. The disks are positioned at the contact points.

Model of the rigid body system

Rigid body systems in physics consist of *rigid bodies* and (generally elastic) *connections*. The bodies have a *center of gravity* and any number of *link points* defined in the body's local coordinate system. The center of gravity is by default represented by a cube which also indicates the orientation of the body. Optionally, the center of gravity can be represented by a sphere, too. The orientation is also indicated by drawing the body's local coordinate system as shown in Figure 6.2(b) and (c). This is especially necessary when the user decides to use a sphere glyph to represent the center of gravity.

Connections run between link points of two bodies. Each link point has a *contour* assigned which is used in the simulator to calculate contact points between bodies. This modeling paradigm implies that, for example, a sprocket is modeled as a single rigid body with one link point at its center. The contour of this link point (shown in green in Figure 6.2(a)) is the actual shape of the sprocket. Similarly, each chain link has link points at both of its pins that have circular contours, shown in blue. The smaller circle is the roller's surface that comes into contact with the sprocket's teeth while the larger circle is the outer surface of the link sliding on the guides.

The actual shape of the bodies is represented by their contours. The perception of space and the general aesthetical value of this simplified representation are limited. We can significantly improve the situation by showing 3D models. Although there is no actual 3D simulation geometry associated with the bodies, the contours can be extruded to create a better impression, as shown in Figure 6.9

Scalar attributes

A colored disk is a very simple glyph that can represent a scalar value by mapping it to color. Many of the simulation results are not scalars, but vectors. We offer two options to reduce them to scalars so that they can be visualized using colored disks. Either the length of the vector can be displayed by color, or one of the vector's components. Despite their simplicity, colored disks have been found to be especially useful in visualizing magnitudes of forces. They provide a quick and simple way of locating the areas of extreme forces; something engineers perform very often as a first step when investigating chain drives. The colored disks in Figure 6.3 are drawn at the contact points between chain links and the sprocket and they visualize the contact forces. There is one disk at each contact point.

Additional attributes could potentially be encoded in the diameter of the disks, but we do not do that. The reason is the typical way engineers use the colored disk glyphs. They view the entire chain and look for extreme values. There are disks at each chain link, and they appear relatively small when the entire chain is visible. Their diameter is definitely too small to be perceivably modulated. Therefore, the sizes of all disks are the same.

We offer three options to map numbers to colors to support different analysis situations. We refer to them as global, local, and user defined scaling. *Global scaling* maps the minimum and maximum of the selected data attribute over all simulation time steps to the first and last color in the color gradient. This preserves the consistency of the visualization during animation. The actual range of result values in the current time step can be significantly smaller than the global range. There are often transient phenomena in the simulation that have a profound impact on the global extreme values, but they are present only in a few specific time steps. In all other time steps, the global color mapping uses only a small fraction of the color gradient, thereby impairing the resolution of the visualization. *Local scaling* maps the minimum and maximum of the selected attribute in the current time steps to the color gradient. This provides better resolution for the more detailed analysis of the currently displayed time step, at the cost of loosing temporal coherence. Finally, the mapped range can be *user defined*, which allows the analyst to "zoom in" on a specific range of result values. The values below the specified minimum and above the specified maximum are displayed using two dedicated colors to indicate any clipping.

Vector attributes

Probably the most straightforward way to visualize vector quantities is using arrows. Force, translational velocity and acceleration are often represented by arrows in physics, thus it comes natural to use them in our visualization framework, too. The direction and the magnitude of the visualized attribute are indicated by the direction and length of the arrow. The starting point of the arrow identifies the point in space where the vector acts.

We offer 4 different drawing styles of arrow glyphs (flat head, cross head, 3D cone head, and pyramid-shaped variants), each with several customizable parameters, shown in Figure 6.4. One can use different drawing styles for the simultaneous visualization of different vector attributes and avoid confusion. It is possible to visualize the complete 3D vector or only its components with respect to the body's local coordinate system. For example, the normal component of the

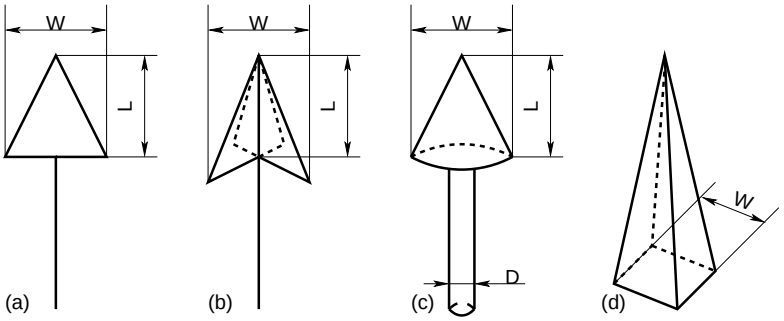


Figure 6.4: Models of (a) flat head, (b) cross head, (c) cone head and (d) pyramid-shaped arrows. Customizable parameters (width and length of arrow head, diameter of shaft) are also indicated.

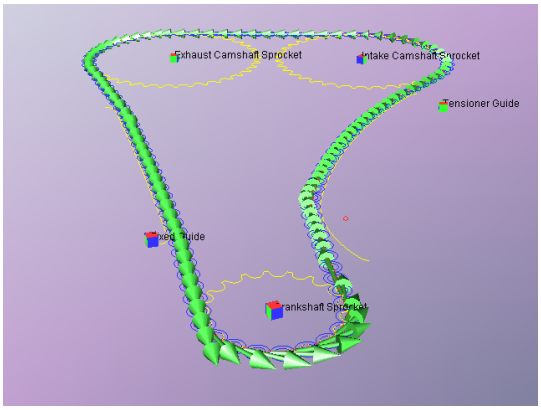


Figure 6.5: Tangential velocities of chain links. Arrows occlude each other when drawn in their actual direction.

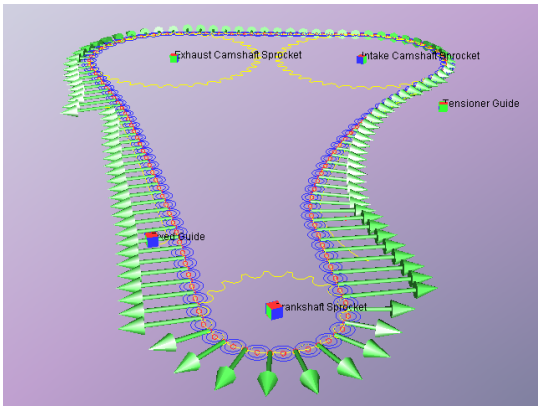


Figure 6.6: Tangential velocities of chain links drawn normal to the actual direction. Compared to Figure 6.5, occlusion is reduced.

chain links' velocity is of great interest to engineers, because it is directly related to the chain's vibration. If arrows display that component only, then the visualization can highlight this feature of interest. An example is shown in Figure 6.12.

In addition, it is also possible to show any component along *any* axis of the local coordinate system. The advantage of this becomes obvious when one tries to display the tangential velocities of chain links. The arrows would "pile up", occlude each other and render the image illegible, as shown in Figure 6.5. We propose to display the tangential component *normal* to the chain itself, as shown in Figure 6.6. This creates a useful visualization of the magnitude while the direction is known to be tangential. It must be mentioned that novice users can easily misinterpret such visualizations. However, experienced mechanical engineers are used to such representations because it has been a common method of displaying chain forces in figures [180].

Either the length of the arrow glyph can be changed to visualize the assigned attribute, or one can scale the entire arrow glyph in 3D. Although the latter introduces the "visualization lie" [251] effect because of the nonlinear change in surface, in some scenarios it still turns out

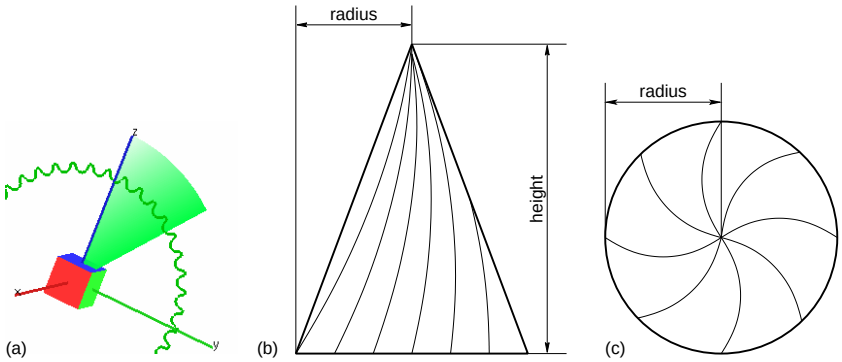


Figure 6.7: Glyphs for angular attributes. (a) sector glyph on one body visualizing angular velocity. (b), (c) sketch of a spiral mapped on the side of a cone. (b) side view, (c) top view.

to be preferred by engineers because it makes extremely large attributes stand out even more. The mapping from data attribute to geometric length follows a scheme similar to that of the color mapping of colored disks. The data range mapped to the arrow size can be global, local, or user defined. It is also possible to apply optional color mapping to the arrows to reflect their magnitude. Annotations displaying actual numeric values can be attached to the arrow glyphs to support quantitative analysis [216]. This is also illustrated in Figure 6.12.

Angular attributes

The visualization of angular velocity and acceleration is somewhat more challenging. In computer graphics, orientation and angular velocity are commonly described by the axis of rotation (a vector) and the angle of rotation (a scalar) around that axis. To the contrary, the rigid body dynamics simulator we worked with represents the rotation with Euler angles, and the engineers expect to see those three angles in the visual analysis, too.

Inventing a glyph that can visualize three rotational degrees of freedom in an intuitive and unambiguous way is not easy. Therefore, we have opted to visualize only one component of the angular velocity in the body's local coordinate system with one glyph. Additional glyphs of the same type can be instantiated to visualize further components.

Sectors One glyph that has proven to be intuitive is a sector. The sector's center is at the center of gravity of the body. The axis of rotation is indicated by rotating the sector glyph such, that its normal coincides with axis of rotation. The magnitude is visualized by the angle of the sector. The direction of the rotation (clockwise vs. counter-clockwise) is made clear by changing the transparency along the arc. That intuitively indicates the direction of the movement since it gives the impression of an arm leaving traces while sweeping the arc. Figure 6.7(a) shows a single body (the center of gravity of a sprocket) and a sector glyph that indicates its angular velocity. The mapping from simulation results to sector angles can also follow the previously mentioned global, local, or user defined schemes. As with arrows, magnitudes can also be optionally mapped to color and actual values can be displayed as annotations.

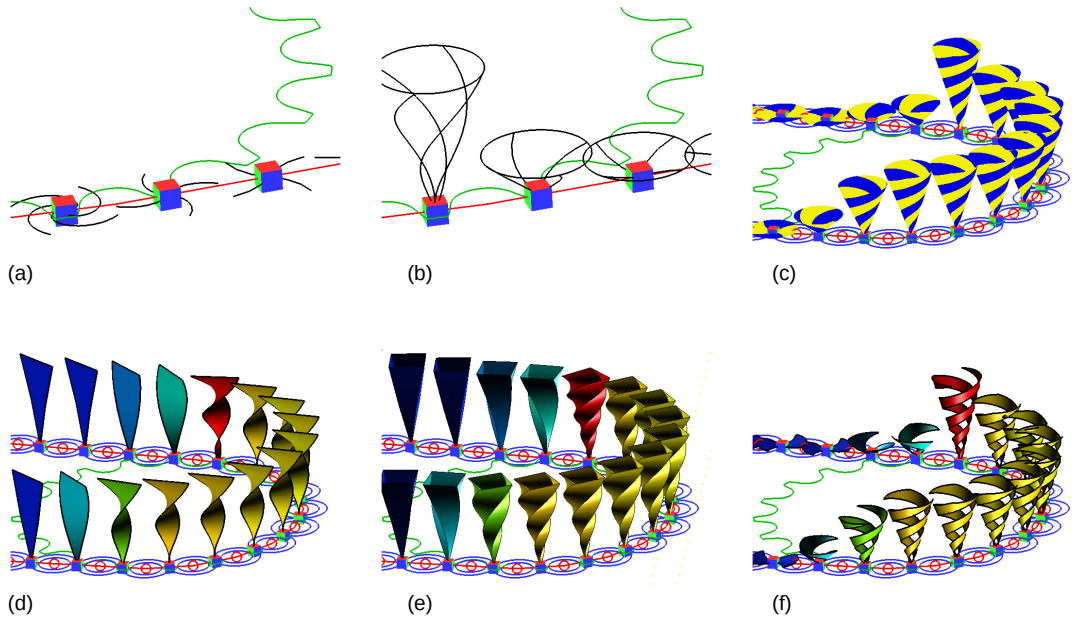


Figure 6.8: Different ways to draw the spirals: (a) wireframe, (b) wireframe with circle, (c) filled, (d) helix, (e) pyramid, (f) ribbon.

Spirals The sector glyph is quite intuitive, but it has a serious shortcoming. It is a flat, two dimensional object, and when viewed under very small angles it is difficult to interpret. In the extreme case it appears as thin lines on the screen and it is completely unable to convey any information other than perhaps the color.

To overcome this problem we propose a glyph that can reliably visualize angular attributes independent from the view direction. The basic idea, illustrated in Figure 6.7(b) and (c) is simple: take a star-shaped glyph with a few arms and transform it into a spiral by twisting its center. The larger the attribute the more the spiral is twisted. This object is still 2D, therefore it is prone to similar visibility problems as the sectors. Now map these lines onto the surface of a cone and make the cone's height also a function of the attribute being visualized. This results in a 3D glyph that can visualize angular attributes reliably regardless of the viewing angle. When viewed from the top or bottom, the twisted spiral arms are easily interpreted. When seen from the side, the spiral arms are not as clearly visible as from the top, but in this case the height of the cone provides additional clues. The cone is rotated so that its axis coincides with the axis of rotation. The direction of the rotation is indicated by twisting the spiral clockwise or counter clockwise, respectively.

The user can change (1) the maximum twist angle, (2) the radius of the disk at the base of the cone, (3) the number of spiral arms, and (4) the scaling factor of the cone's height. To facilitate the simultaneous visualization of many attributes, we offer six variants, shown in Figure 6.8. The scaling of the cones' height can be disabled, so that only the twisting indicates the rotation (see (d) and (e) in Figure 6.8), or they can be made flat, as in (a). The previously mentioned global,

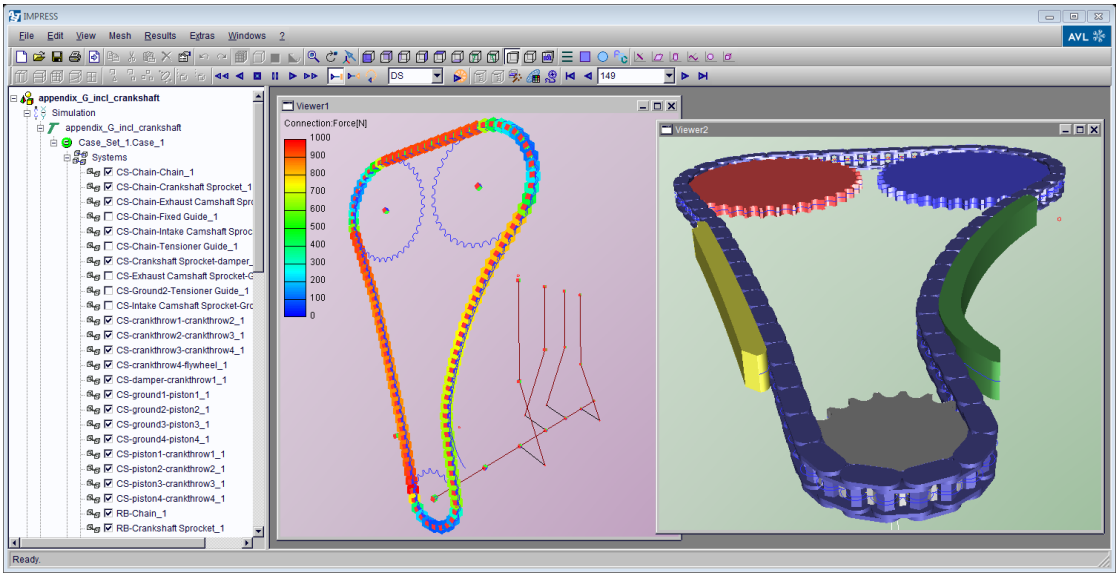


Figure 6.9: Interactive 3D visualization of chain drives. The two views show the same model. The 3D geometry in the right hand side window is created by extruding the contours.

local or user defined scaling schemes can be used to map numbers to the amount of twist. Color mapping and annotations are again optional.

To further improve the perception of rotation and angular attributes, the spirals can optionally be animated. The speed and direction of the rotation reflects the attribute visualized. This feature has been surprisingly popular with users, perhaps, because it indicates rotation in the most natural way, although the assessment of the rotational velocity in such animated visualization may not be very accurate.

6.2.3 Application Example

The glyph-based 3D visualization has been integrated into IMPRESS, a commercial application developed at AVL-List GmbH, originally for the interactive 3D visual analysis of computational fluid dynamics simulations [146]. Integrating the approach described in this section into the application added the capability of analyzing rigid multibody systems. Mechanical engineers use it regularly to analyze chain and belt driven timing drives in engines. In this section we present a demonstration of the visual analysis procedures.

Integrating rigid body visualization into IMPRESS

A snapshot of the application is shown in Figure 6.9. The application window consists of a menu bar, several toolbars, a tree view of the model and, finally, one or more 3D views in the middle. Model elements (sprockets, tensioners, etc.) are listed in the tree view on the left. Groups of bodies, e.g., bodies that belong to one chain are shown as one entity. These groups are referred to as *systems*.

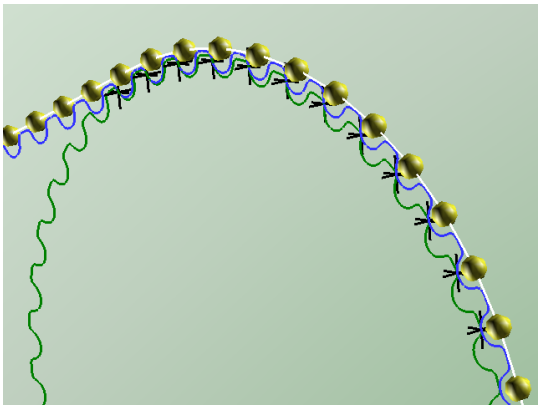


Figure 6.10: This figure captures the moment when a toothed belt skips a tooth of the sprocket. Black crosses indicate the contact points. Note that the belt does not properly mesh with the sprocket, but only the teeth come into contact with the tips of the cogs.

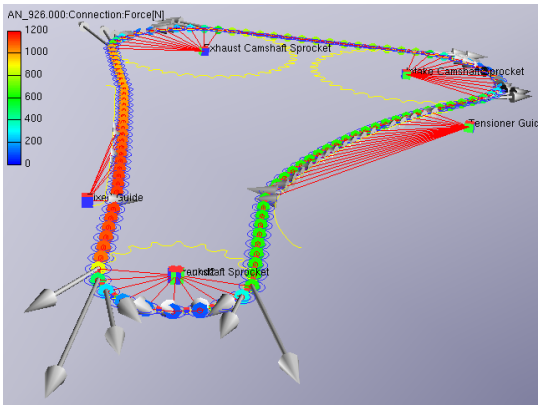


Figure 6.11: Connection forces between chain links (colored disks) and contact forces between chain links and the sprockets and guides (arrows). This figure also demonstrates creating several visualization objects simultaneously.

The model is displayed in 3D views. The view can be panned, rotated and zoomed as desired. The components of the systems (center of gravity symbols, link points, etc. in Figure 6.2) can be shown or hidden and their colors and sizes can be customized. It is possible to hide or drag groups that are temporarily out of interest in order to reduce occlusion. Dragged objects can be returned to their original position by a single click.

In order to view attributes of bodies or connections in a system, the user selects the system and clicks the icon “Rigid body results”. A *visualization object* consisting of a group of glyphs is created. The visualization object contains one glyph for each body or connection in the system. A dialog pops up where the user can select the attribute and glyph type as well as customize glyph parameters and color mapping options. Any number of these visualization objects can be created for each system, and they can be hidden or dragged just like the systems themselves.

Any number of 3D viewing windows can be opened, displaying the same or different simulation runs. The model can be animated using VCR-like controls, enabling the user also to single-step or fast-forward on the time axis. If data are not available for a point on the time axis, then the system can interpolate them from the adjacent time steps. Viewing windows can be linked to animate together, also when they display different simulation runs, to facilitate their comparison. The system is fully interactive during animation, too. New visualization objects can be created while the display is animated. There is a movie director module to facilitate the creation of movies for presentations. Events such as show/hide objects, change the properties of the mapping from data attributes to glyphs, zoom in/out, rotate view, etc. can be added to the timeline. All settings are saved in a project file to allow the users to continue exactly where they left off.

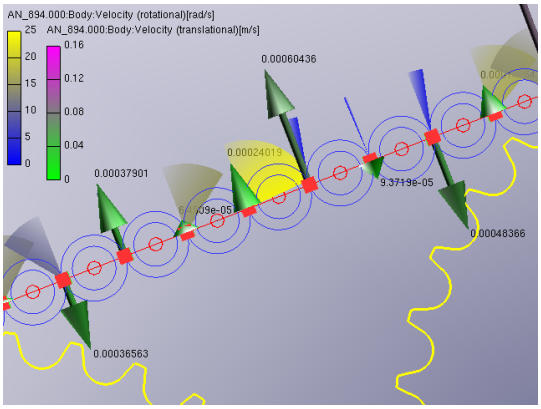


Figure 6.12: Rotational velocity of chain links visualized using the sector glyph. Annotated arrows show the normal component of the translational velocity. Vibration is clearly seen.

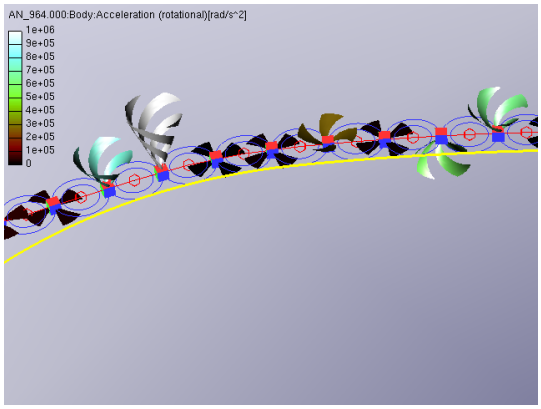


Figure 6.13: Rotational acceleration depicted by spiral glyphs. Opposing directions of rotation are indicated by opposite twist directions and different orientation of the cone, too.

Interactive visual analysis workflow

The engineers typically start the analysis by examining if the simulation was correctly set up. They run the animation looking for obvious signs of incorrect initial conditions. Mismatching chain and sprocket contours, sprockets defined at the wrong position or in the wrong orientation are very easy to detect. With toothed belt drives, loose tensioners may allow the belt to skip a tooth of the pulley, which is also easily seen in the animation (see Figure 6.10). This is a serious problem and it can potentially cause the complete destruction of the engine. These modeling errors are very difficult to discover without 3D visualization.

The next step of the analysis is examining forces along the chain drive, looking for areas of extremely high forces. These areas are quickly found by creating colored disks that show connection forces between chain links and then running the animation. In Figure 6.11 the span of the chain on the left entering the sprocket is easily identified as being under heavy tension. Engineers are interested in the contact positions and contact forces of the chain links and the sprockets. This information is valuable in order to understand which areas will be subjected to extreme wear. In Figure 6.11 the location and direction of the forces are seen which would be quite difficult without 3D visualization.

The vibrational motion of chain links in long spans is of concern to the engineer. This can be visualized by showing the translational and angular velocities of chain links as seen in Figure 6.12. The relevant component of the vibration is the one normal to the chain links' general direction of travel. The vibration is clearly visible in Figure 6.12, because the arrows show only the relevant normal component.

Rotational acceleration and torque can be visualized using the spirals introduced in Section 6.2.2. Figure 6.13 shows a close-up of rotational acceleration of chain links. It is obvious in this figure that the chain links undergo abrupt acceleration as they come into contact with the

yellow tensioner guide. This can be an indication to the engineer to change the profile of the tensioner to allow a smoother contact. Naturally, this can reduce noise levels and extend the chain's expected life span.

We must mention that not all color scales shown here are perceptually uniform [30, 151, 149]. The rainbow color map in Figure 6.11, for instance, has a large section of shades green that are hard to distinguish. Actually, this is not a disadvantage in this application context, because users are generally more interested in the extremely small or large magnitudes.

6.3 Interactive 3D Visual Analysis of Elastic Body Dynamics

In this section we describe tools for the interactive 3D visual analysis of elastic body simulations. The framework includes glyphs similar to those used in the previous section. Elastic bodies, unlike rigid bodies, do have actual 3D meshes associated with them. Therefore, we also propose methods for the visualization of simulation results directly in the context of those meshes. The methods described in this section have been integrated into the same application as the rigid body visualization. As detailed in Section 6.2.3, several 3D views can be opened, the views can be panned, rotated and zoomed, several visualization objects can be created to depict simulation results, and animations and movies can be created.

It is possible to jointly analyze rigid and elastic body simulations. This is very relevant, because the motion of certain engine components (e.g., timing chain) is computed using rigid body simulation, while the motion of other components (e.g., the crankshaft) is computed using elastic body simulation. The components of the same engine are simulated using different principles. The two simulation models can be coupled during computation to represent the interaction between engine components. It is very important that the resulting multi-model [118] data set can be jointly visualized and analyzed in the same 3D view.

6.3.1 Simulation of Elastic Body Systems

Many components of internal combustion engines suffer significant deformations when the engine is running. Conrods bend under load during the combustion, crankshafts endure bending and torsional deformation, and the engine block itself vibrates as well. The amplitude of the deformation is of course not very large, but the deformation reduces the durability and contributes to the emission of noise. The deformation can be computed using a simulation model based on the theory of elastic multibody systems. This concept is often used in engineering in fatigue and stress analysis [154] as well as in engine noise analysis [188]. The simulation model consists of two fundamentally different types of entities: linear elastic *bodies* that represent engine components (crankshaft, conrods, pistons, etc.); and *joints* (bearings, dampers, etc.) that connect the bodies and often feature non-linear behavior.

Bodies are defined via finite element (FE) meshes that are built from 3D CAD models. The FE mesh of an engine block, for example, may consist of several hundred thousands of elements. Each node has three translational and three rotational degrees of freedom. Running forced response analysis in the time domain on a structure with so many degrees of freedom

is not feasible on current hardware. Therefore, the meshes go through a condensation process before simulation.

The *condensation process* [229] creates a simplified, "condensed" mesh. It chooses a subset of the nodes (typically not more than a few hundred) of the mesh that represent the body in the simulation. The nodes are identified by *node IDs* (numbers). The engineer can also manually mark degrees of freedom of specific nodes that are to be preserved in the condensed model. These nodes are called *retained static degrees of freedom*. *Single point constraints* can be defined to disable one or more degrees of freedom of specific nodes. *Multiple point constraints* establish a relationship between degrees of freedom of a group of nodes. The condensation also creates mass, stiffness and damping matrices that describe the dynamic properties assigned to the nodes in the condensed mesh. The resulting reduced model and the matrices define the bodies in the simulator. The simulator output is directly available for the condensed mesh nodes only. The data can be *recovered* for the entire FE mesh using the information from the condensation process. It is important to understand that the simulator works only with the condensed meshes, not with original FE meshes. The result is a much faster simulation which produces results that are still valid for the entire structure, provided the condensation was correct. FE modeling and condensation is performed in third party software (Abaqus [2], MSC.Nastran[3], etc.).

Joints establish connections between bodies and restrict their motion. A joint connects groups of nodes in two bodies. The nodes are referenced by their node IDs, for example: "connect node 2 of the crankshaft with nodes 10,11,12,13 of the block". Alternatively, named *node sets* containing several nodes can be defined and used in the specification of joints. For example, the nodes of the engine block that can come in contact with the crankshaft can be listed in sets called "MainBearing1", "MainBearing2", and so forth, for each main bearing, respectively. There is a utility that can automatically identify the nodes that constitute main bearings.

After all bodies and joints are created, additional information about the crank train, external loads and boundary conditions, such as lubricant type are defined, and then the simulation is run. The simulation is usually run for 3–4 engine cycles which equal 6–8 complete revolutions of the crankshaft to allow the initial transients to settle. Depending on model's complexity and the length and the resolution of the simulated time interval this may take several hours for a single simulation case. The computed motion of the bodies consists of two components: (1) the global motion, and (2) the local displacement of nodes. About 10 additional attributes are computed for each node, including load and inertia forces, as well as shear rate and stress. The interaction of the crankshaft's overall rotation with the vibration of its parts is taken into account. The resulting non-linear inertia forces like gyroscopic effects are also handled in the simulator.

6.3.2 3D Visualization of Elastic Multibody Systems

The visualization of elastic multibody systems involves displaying the meshes of the bodies enhanced with several different glyphs to indicate features like joints and constraints. Occlusion is generally a major issue in this visualization; hence we propose possible techniques to alleviate the problem.

[2]ABAQUS Inc. `http://www.simulia.com`

[3]MSC.Software Corp. `http://www.mscsoftware.com`

Bodies

Bodies are the structural components of the model. Each body has a local coordinate system which is also indicated by an axis triad. The axis of rotation is highlighted in a different color. Each body can have two associated meshes: the FE mesh and the condensed one.

FE mesh This is the FE mesh that was used as input for the mesh condensation process. Surface and/or wireframe FE meshes can be shown. Occlusion is a major issue in this context, since virtually everything (crankshaft, conrods, etc.) that is of interest to the engineer is contained. Furthermore, joints usually represent contacts between bodies thus these bodies almost always obscure them. We offer four means of reducing occlusion in FE meshes:

- The outermost layers of FE meshes can be removed one-by-one temporarily, much like peeling an onion. This allows investigation of the inner elements.
- An arbitrary clipping plane can be applied to each FE mesh. The elements (cells) in one half-space are removed. This can be used to remove half of the engine block, for instance, creating an exploded view of the engine. The plane can be freely rotated and dragged. Unlike typical clipping planes defined in OpenGL or other libraries, this plane clips complete elements of the FE mesh. An element (cell) is either completely visible or completely clipped, but never cut in half, thus the notion of elements in the mesh is better preserved.
- Similarly, an arbitrary intersection plane can also be used. Only elements that intersect the plane are shown. As with the clipping plane, complete elements are preserved.
- Arbitrary groups of FE elements called *cell sets* can exist. Those sets often represent semantic parts of the mesh; for instance, the left side of the engine block. They can be hidden or rendered in different colors. Cell sets in FE mesh files are read automatically. In addition, there are several powerful ways to define and edit sets.

Transparency can be combined with any of the above tools. Figure 6.14 demonstrates these methods in order to provide a good view of the whole engine. Half of the block is clipped away and the crankshaft is drawn semi-transparently.

Special node constraints are represented by small wireframe glyphs attached to the nodes (see Figure 6.15). Retained static degrees of freedom are indicated by a small coordinate system icon. For each enabled translational degree of freedom, the corresponding axis of the coordinate system is shown. Rotational degrees of freedom are indicated by small arcs around the respective axes. Single point constraints are depicted by the triangular “constraint” icon commonly used in mechanical engineering software. Multiple point constraints are displayed by connecting the constrained nodes.

Condensed mesh As described in Section 6.3.1, this is a reduced mesh, consisting of nodes only. The visualization simply shows the nodes with optional node ID labels and line segments connecting the nodes. Figure 6.16 shows the reduced mesh of a crankshaft.

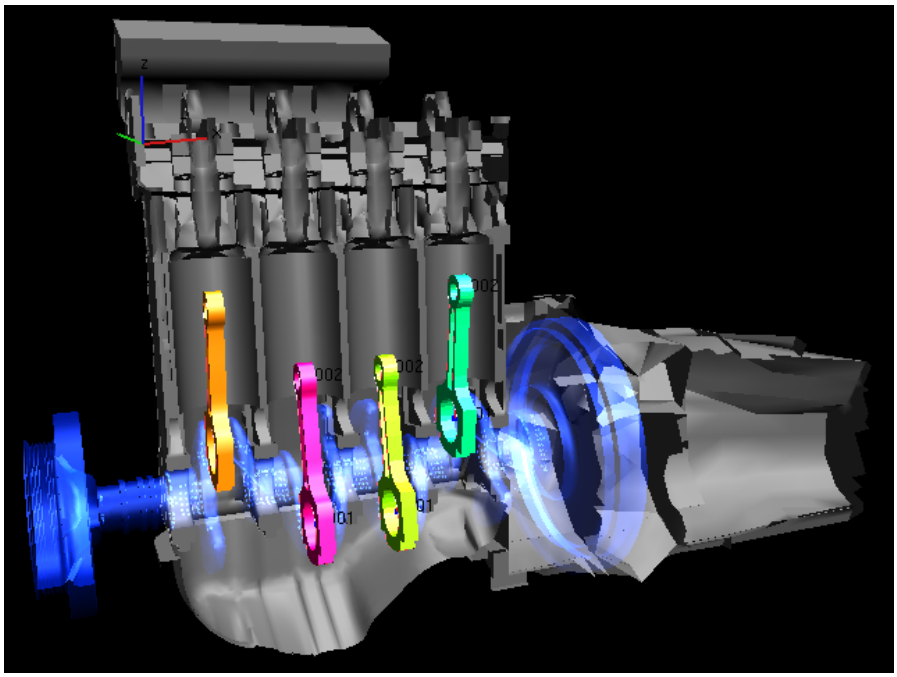


Figure 6.14: 3D view of a 4 cylinder inline engine. Half of the engine block is clipped to make the inside visible.

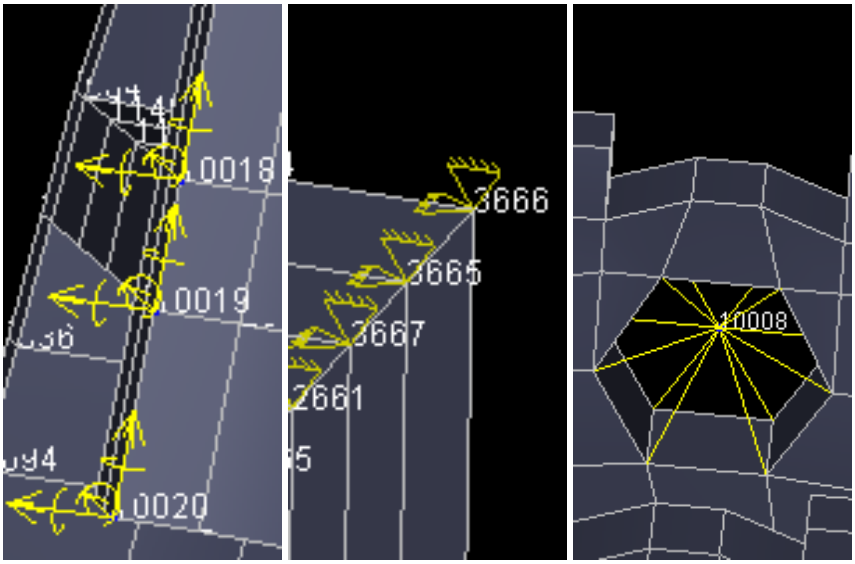


Figure 6.15: Close-up views of an engine block with glyphs representing retained static degrees of freedom (left), single point constraints (middle), and multiple point constraints (right). Labels show node IDs.

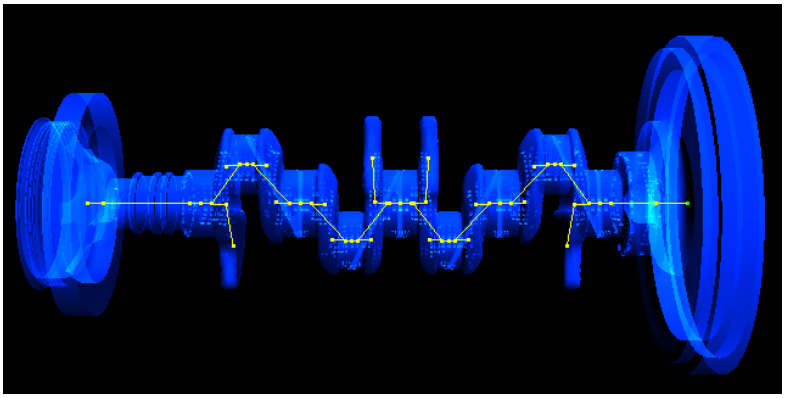


Figure 6.16: Condensed mesh of the crankshaft used in the simulator. The transparent FE mesh is also shown as context.

Joints

It is worth mentioning that joints cannot be represented as easily as bodies because they differ fundamentally: there is no mesh associated with them. They merely describe the contacts between nodes of meshes. There are over 20 different joint types in the elastic body dynamics simulator we used. In the 3D view they are represented by lines that connect the actual nodes of the bodies. We also show different icons for each joint type, the same small bitmaps that are used elsewhere in the simulator's user interface to represent joints. For some specific types (bearings, piston liners, etc.) 3D glyphs have been incorporated. A main bearing is shown in Figure 6.17(a). In this image the ring is a glyph that indicates the diameter and the width of the bearing. In Figure 6.17(b), a piston-liner guidance is shown. The beams indicate how nodes of the engine block are connected to the piston pin for simulation. The yellow labels show IDs of the connected nodes.

6.3.3 Visualization of Simulation Results

The simulation computes results for the condensed mesh. The results on the condensed mesh nodes can be visualized by glyphs placed on the nodes. Several visualization objects consisting of glyphs can be created, analogous to the rigid body visualization detailed in Section 6.2.3. The results for the original FE mesh can be recovered using the information generated during the condensation. Data that are recovered to the FE mesh can also be visualized directly on the mesh.

Deformation of bodies

At each simulation time step t, simulation result data contain the global motion $\mathbf{g}(t)$ of the body and the local displacement component $\mathbf{d}_i(t)$ of each node. The complete motion $\mathbf{m}_i(t)$ of each node is the sum of the two, $\mathbf{m}_i(t) = \mathbf{g}(t) + \mathbf{d}_i(t)$. The basic approach is to produce an animated visualization of the elastic body's motion by displacing its nodes by $\mathbf{m}_i(t)$ for each time step t.

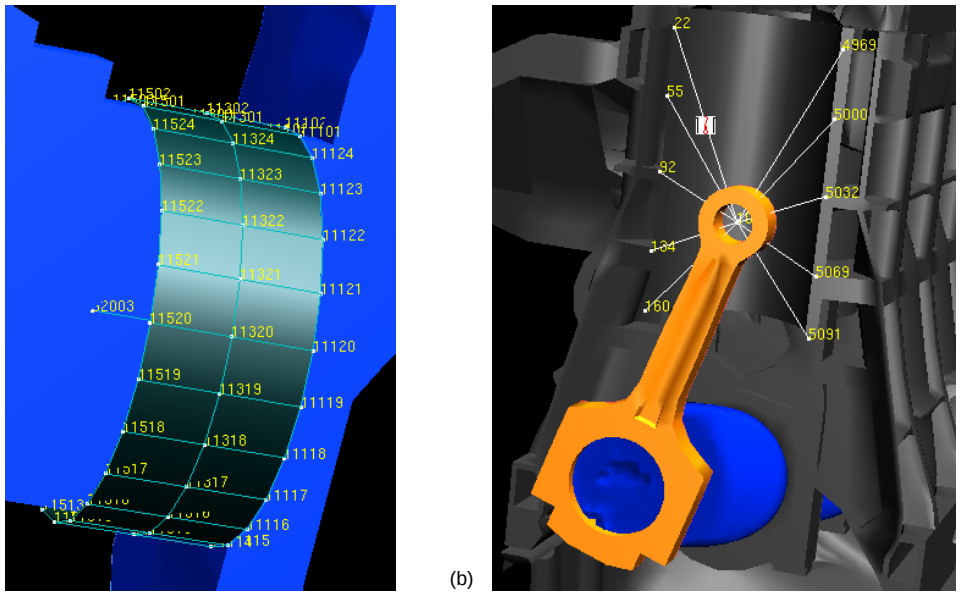


Figure 6.17: (a) One of the main bearings of the engine. Part of the crankshaft is clipped to reduce occlusion. The IDs of the connected nodes are shown. (b) Piston-liner contact. The piston itself is hidden to reveal the connection.

The deformation is generally very small. The displacement of nodes is often smaller than a pixel. The deformation can be made more visible when scaled up by a factor c so the motion of nodes is $\mathbf{m}_i(t) = \mathbf{g}(t) + c \cdot \mathbf{d}_i(t)$. Note that this magnified deformation is not physically correct. It is merely a geometric magnification to make deformation more visible. The global motion (overall rotation) of rotating bodies (e.g., crankshafts) can be turned off and only the vibration component shown to allow exact comparison of the crankshaft's deformation at various crank angles.

When deformation of rotating components (e.g., crankshafts) is analyzed, the torsional deformation is often of special interest. The linear magnification of displacement can create very strange visualizations. Parts of the body are distorted in a very disproportional way, and the torsional deformation, which is often the interesting feature, is very difficult to discern. Some of the problems associated with this approach are shown in Figure 6.18.

The situation can be improved by magnifying the displacements in a cylindrical coordinate system defined by the body's center of gravity and axis of rotation. Different scaling factors can be used for the radial, angular and axial component of the deformation. If the radial and axial scaling factors are set to zero and the angular scaling factor is not zero, then only the angular component of the deformation is preserved. The resulting visualization, shown in Figure 6.19, better depicts the torsional deformation. We must again note that this is not a physically correct deformation, but merely a method of making torsional deformation more visible. The magni-

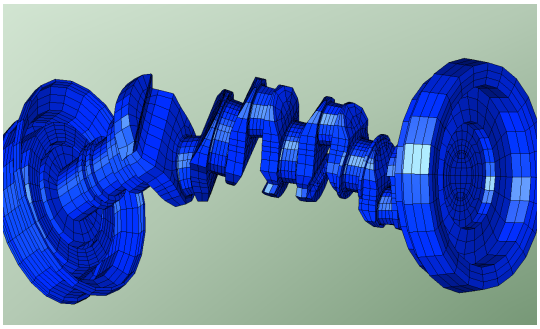


Figure 6.18: Deformation of a crankshaft scaled up 500 times. Note the disproportional distortion of the flywheel and the crank webs. Compare this to the original shape as shown in Figure 6.16.

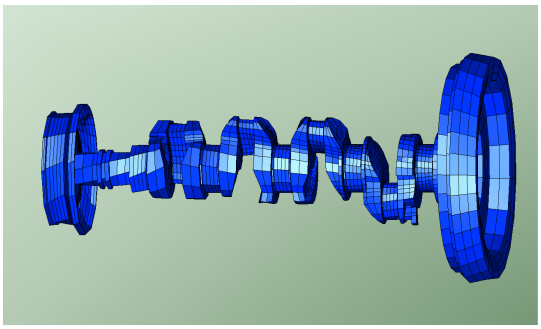


Figure 6.19: The same crankshaft as in Figure 6.18 with only the angular component of the deformation scaled up 500 times. Torsional vibration is more visible.

fication of the deformation in the cylindrical coordinate system is described by the following algorithm:

for each node i in the mesh
  $\mathbf{p}$ = original local coordinates of node i in the mesh
  $\mathbf{q} = \mathbf{p} + \mathbf{d}_i(t)$
  (r_p, ϕ_p, z_p) = cartesian_to_cylindrical($\mathbf{p}$)
  (r_q, ϕ_q, z_q) = cartesian_to_cylindrical($\mathbf{q}$)
  $r_s = r_p + c_{radial}(r_q - r_p)$
  $\phi_s = \phi_p + c_{angular}(\phi_q - \phi_p)$
  $z_s = z_p + c_{axial}(z_q - z_p)$
  $\mathbf{s}$ = cylindrical_to_cartesian(r_s, ϕ_s, z_s)
  $\mathbf{f} = \mathbf{p} + \mathbf{s} + \mathbf{g}(t)$
  move node i to position $\mathbf{f}$

Surfaces and cross-sections

Attributes can be displayed using color mapping on the surface of the FE mesh. For example, in Figure 6.20 the displacement of the surface of the engine block is shown. Parts of the mesh can be hidden by the tools described in Section 6.3.2 so that data pertaining to cells inside the body can be made visible. The global, local, and user-defined color mapping schemes introduced in Section 6.2.2 can be used to support different analysis tasks. When parts of the mesh are hidden, then the color mapping can use the range pertaining to the visible part of the mesh. Analogous to coloring the mesh surface, planar cross-sections of the meshes can be created and colored to display attributes of the cells that the plane intersects. The intersection plane can be rotated and translated in the volume as a probe. The cross-sections can also be used as the reference plane of height fields (shown in red in Figure 6.21). The height field can depict any scalar attribute, including components and magnitudes of vectors.

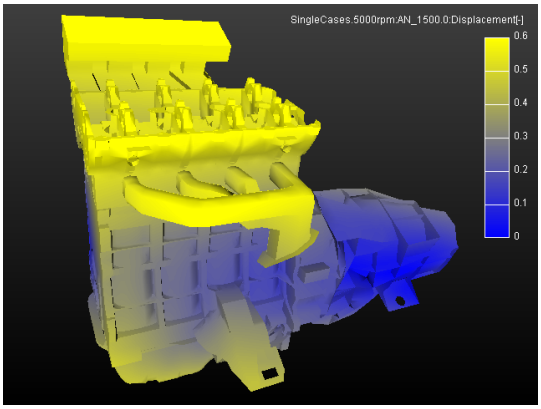


Figure 6.20: Displacement of the surface of the engine block shown using color mapping.

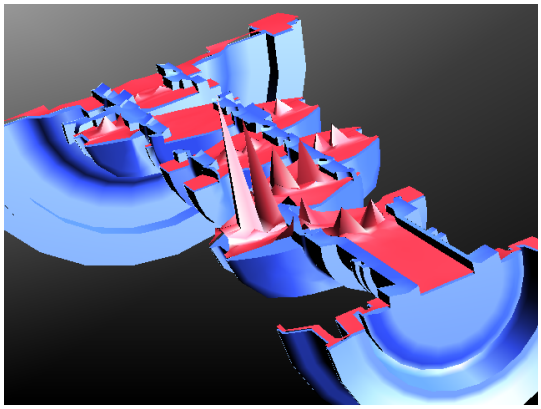


Figure 6.21: Stresses on the cross-section of the crankshaft visualized using a height field. Peaks highlight areas of high stress.

Glyphs

Many of the simulation results' attributes are vectors. Displacement, velocity, acceleration, and forces can be visualized very intuitively by arrow glyphs, similar to those used for the visualization of rigid multibody systems (Section 6.2). Arrows are positioned at the nodes of the condensed mesh. In Figure 6.22 the arrows show the constraint forces on the crankshaft. Furthermore, vector attributes can be visualized by drawing arrows at each cell of the surface or the cross-section, similar to the hedgehog visualization [226] used in computational fluid dynamics. Unfortunately, this often leads to a very crowded display when the mesh is of high resolution. In order to reduce the visual clutter and occlusion, the resampling tool proposed by Laramee [145] can be adapted. As an illustration, note that there are fewer arrows than nodes indicated by the wireframe in Figure 6.23. Drawing arrows at each node would only clutter the view.

6.4 Evaluation

Before the introduction of the 3D glyph-based visualization, engineers at AVL-List GmbH used static diagrams to evaluate and analyze timing drive simulation results. Experienced engineers can certainly infer useful information from diagrams. However, they still face difficulties understanding position and orientation of bodies as well as directions in 3D space. It is especially difficult to judge and compare relative positions and distances. Characteristics of temporal behavior, such as frequency of vibration are also intuitively visualized using animation. Engineers using the application listed the following advantages to their previous workflow:

- Contours that are in contact and the actual contact point positions are clearly identified. This is especially important if the model is somehow incorrect, for example the contour of a sprocket is not continuous or it does not match the contour of the toothed belt. Discovering a very loose

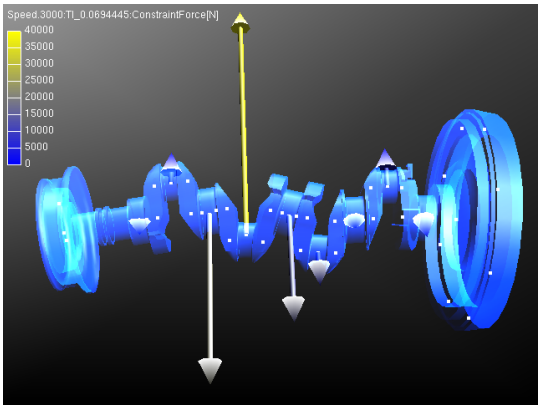


Figure 6.22: Distortion of the crankshaft under load. Deformation is magnified by two orders of magnitude. The arrows indicate the constraint forces at the nodes of the reduced crankshaft mesh.

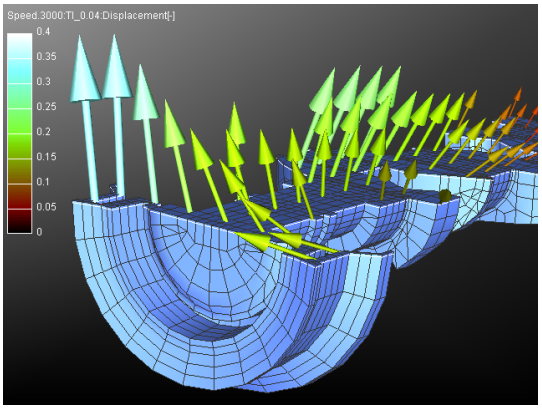


Figure 6.23: Arrows depicting displacement on a cross-section of the crankshaft. The displacement data available at all nodes are resampled to a grid of lower resolution to reduce visual clutter.

toothed belt that skips the teeth of a pulley (see Figure 6.10) is also very easy. Errors in modeling and initial conditions became a lot easier to detect. Finding these types of problems used to be complicated because the irrational simulation results (e.g., extreme forces) were the only indication.

- Directions of forces are made clear. This is important when the engineer expects a force to act in one direction, but simulation proves that it acts in some other direction. In fact showing the direction and magnitude of vector attributes at the same time is something they could not do before without 3D visualization.

- Motion of tensioners can be seen clearly. This has also been very difficult to understand by studying diagrams.

- Animation shows all chain links at a given time step creating an overall impression of how the chain moves. To the contrary, individual charts were created previously for each chain link and the overall motion was difficult to infer from them.

- It is easier to spot periodic behavior at a given point in space.

- Distribution of forces between chain links is seen. Areas of extreme tension are quickly found.

Less experienced engineers and professionals working in neighboring fields (manufacturing, namely) can also profit from the more accessible presentation of the simulation results. Convincing movies for presentations can be generated with little effort, which is also valuable for marketing purposes. Many user groups ranging from inexperienced, young engineers through experienced professionals to marketing department can benefit from interactive 3D visualization.

6.5 Chapter Conclusions

Tasks like exploring the dependency between simulation control parameters and results, or optimization require the simultaneous analysis of many simulation runs. 3D depictions of different simulation runs can be shown in windows arranged side by side, but in practice this only allows the comparison of a few (two to four) cases. The interactive 3D visualization system is not suitable for the analysis and comparison of many simulation runs and it is not meant for that purpose. To the contrary, it is meant as a tool for the detailed analysis of one or a few simulations with complete 3D spatial context. Those few simulation runs that are analyzed in depth in 3D can be identified using the more abstract approaches described in the previous chapters. The 3D view could be integrated into the coordinated multiple views framework introduced in Chapter 3, and show data pertaining to the brushed simulation cases when there are only a few. We admit, however, that this integration has not been implemented yet.

Chapter 7

Demonstration

"Reality is frequently inaccurate."

— Douglas Adams (1952–2001)[1]

The science of visual analytics is very applicable in engineering applications. Simulation and measurement data sets are complex, optimization goals are often conflicting, the trends and dependencies in the data can be indirect. Engineers and designers must make defensible and responsible decisions because design mistakes can have very expensive consequences if shortcomings are discovered during production. The time-to-market for new designs needs to be short, so designers work under time pressure. They also need to communicate their findings to collaborating teams.

In this chapter we present two case studies that document the visual exploration and analysis of real-world simulation data sets from automotive engineering. Section 7.1 contains the analysis of Diesel fuel injection system simulation data. In Section 7.2, an optimization of a timing chain drive is presented. Both case studies were performed working in cooperation with engineers at AVL-List GmbH (`www.avl.com`), the largest privately owned company for the development of powertrain systems with internal combustion engines. The case studies demonstrate that the approach described in the previous chapters is indeed applicable and useful in the analysis of real life engineering problems. All simulation data in this chapter are courtesy of AVL-List GmbH.

7.1 Interactive Visual Analysis of a Fuel Injection System

There are many (often conflicting) goals of Diesel engine design including the need for high power and good fuel efficiency, meeting emission regulations, reducing noise levels, and improving driveability (smooth and reliable delivery of power at various engine speeds). The fuel injection system is the key Diesel engine component to achieve those goals. The following properties are considered important in the fuel injection procedure:

- high injection pressure for good atomization and combustion,

[1] English writer, humourist, and dramatist, best known as the author of *The Hitchhiker's Guide to the Galaxy*.

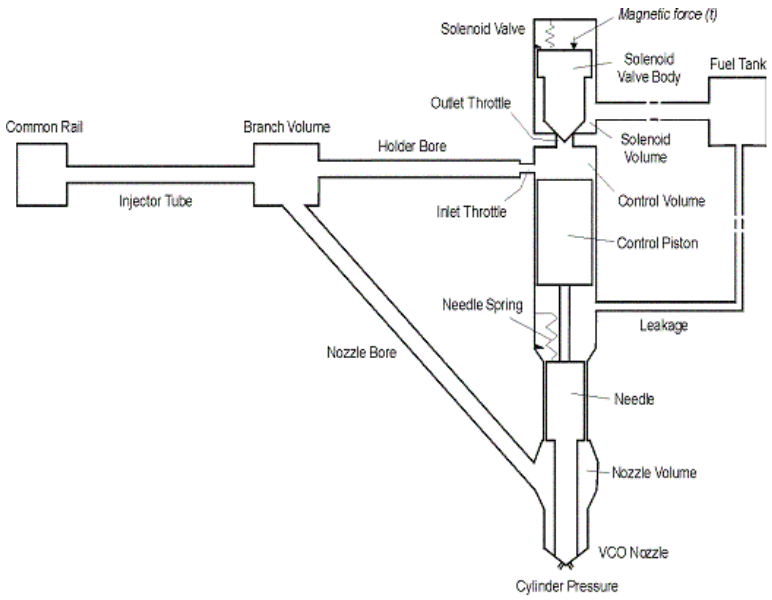


Figure 7.1: Main components of a common rail injector system. *Image courtesy of AVL-List GmbH.*

- flexible timing of the injection,
- short pre-injection before the main burst to reduce combustion noise,
- accurate control of injected fuel quantity,
- ability to inject small amounts of fuel to achieve economical operation and good emission properties.

Currently, the two most popular injection systems for Diesel engines are the electronic unit injector type [46, 82] and the common rail injection systems [29]. Matković et al. [161] have demonstrated the interactive visual analysis of an electronic unit injector. In this section, we analyze a common rail injection system. Common rail injection has been identified as an attractive injection system for Diesel car engines. It is offered by all major car manufacturers today.

7.1.1 Diesel Common Rail Injection Systems

Common rail injection systems can be controlled in a very flexible way. Injection pressure and quantity can be controlled with a high degree of flexibility, multiple fuel injections are possible within one injection cycle, and the time and duration of the injections can be controlled precisely by the engine control unit based on the engine speed and load. These properties are instrumental in meeting current and future very stringent emission regulations. Therefore, common rail injection systems are seen as a very popular option by many manufacturers. In this case study we use the simulation results of a conventional series common rail Diesel fuel injection system [29].

The injector is the central part of an injection system that injects a desired fuel quantity into the cylinder. Figure 7.1 shows a typical injector with main components. The common rail

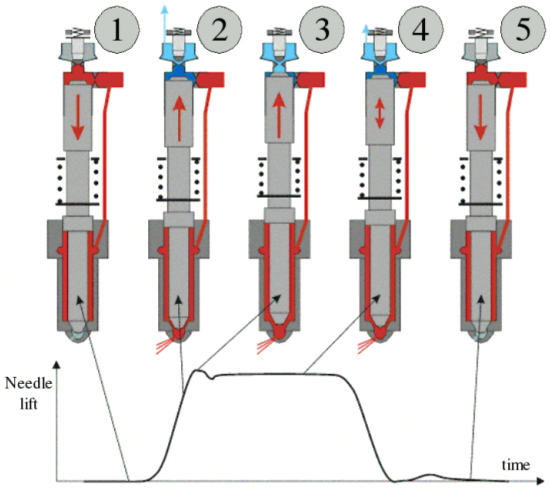


Figure 7.2: Operation of the common rail injector. 1. ECU activates the solenoid valve. 2. The outlet throttle opens. Fuel flow causes a pressure drop in the control chamber. Low pressure results in the needle going up. Injection starts. 3. The needle is open. 4. Injection has the best performances. ECU starts releasing the solenoid valve. 5. The outlet throttle closes. Pressure in the control chamber and the spring force move the needle down. Injection stops.

injector system is controlled by a 2/2 solenoid operated valve. The main components of the injector include:

- needle with the control piston,
- multi-hole VCO nozzle,
- two orifices controlling the pressure in the control chamber and a corresponding control piston (inlet and outlet throttle), and
- solenoid valve body with opening/closing throttle.

The principle of the common rail injector operation is outlined in Figure 7.2. The red, dark blue and light blue colors represent pressure levels. The injection process is initiated when the ECU (Electronic Control Unit) energizes the solenoid valve (Figure 7.2.1). As a result, the outlet throttle opens. Fuel flows through the control volume towards the solenoid volume, and as a result, the pressure in the control volume drops. The pressure difference between the top and bottom sides of the needle pushes the needle up (Figure 7.2.2), at the same time opening the injection nozzle. After some time the needle reaches the stable open position resulting in the optimum injection of fuel into the cylinder (Figure 7.2.3). When the ECU deactivates the solenoid valve, the spring in the solenoid valve closes the outlet throttle in the injector (Figure 7.2.4). As a result, the pressure in the control volume increases. The pressure on the top side of the control piston, together with the nozzle spring force, closes the nozzle and injection process ends (Figure 7.2.5).

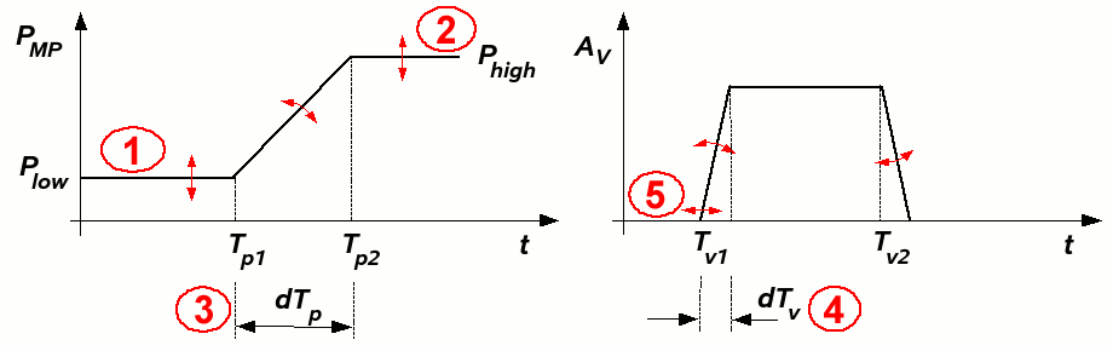


Figure 7.3: Control parameters for the simulation. Left: inlet pressure characteristics are pressure levels P_{low}, P_{high}, and the time interval of pressure increase dT_p. Right: injector valve opening/closing properties are the time T_{v1} when the valve starts to open, and the opening/closing time interval dT_v.

7.1.2 Fuel Injection Simulation

The simulation of the fuel injection process is based on the theory of 1D fluid dynamics and 2D vibrations of multibody systems. Multiple runs (termed *cases*) of the simulation were computed. Each case is represented by its own set of simulation control parameter values. The software can automatically loop the parameters over a specified range and run a simulation variant for each resulting case. The engineers study the resulting output data and attempt to find ideal simulation input parameters for various engine operating situations. This data set follows the model introduced in Section 3.2: I consists of the simulation control parameters and D consists of the simulation output. We now provide a detailed description of the dependent and independent variables.

Simulation control parameters

The injection shape depends mainly on three factors: the nozzle geometry, injection pressure, and timings for valve opening and closing procedures. The influence of control parameters related to nozzle geometry has already been investigated in our previous work [166]. Once (a nearly) optimal nozzle geometry is found and it goes into production, it cannot be changed very often because that would be too expensive. The focus of fuel injection optimization afterwards is usually on varying the remaining two factors.

Therefore, the independent variables in our current investigation are related to injection pressure and injector valve timings only. The injection pressure is controlled by the injection pressure modulation device which is positioned between the rail and injector. In our investigations this device is not modeled in detail, but we take the modulated pressure as input. The characteristics of the pressure on the injector's inlet are described by three independent variables (the red labels 1, 2, and 3 in Figure 7.3). The injector valve actuator that controls the injection timing is described by its opening/closing times and velocities (see the red labels 4 and 5 in Figure 7.3). Although this is a simplified model, it allows the simulation of various types of valve actuators

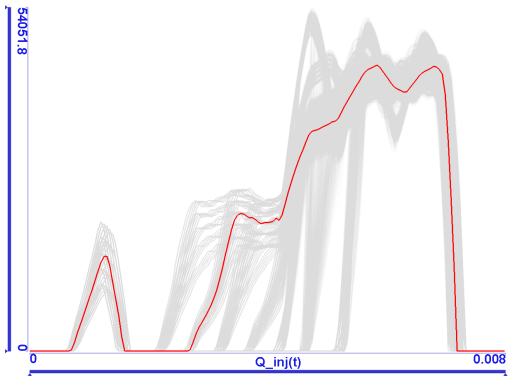


Figure 7.4: A typical shape of the fuel injection rate function graph is highlighted in red. There is a short pilot injection first, followed by the main injection. The function graphs resulting from other combinations of control parameters are gray.

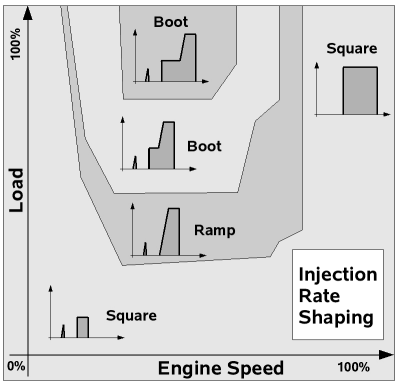


Figure 7.5: Ideal shape of the main injection for various engine operating points defined by engine speed and load. The shape can be classified into three types: square (injection rate steeply increases to a maximum level), ramp (the slope is more gentle), and boot (following a nearly horizontal segment injection rate rapidly increases to maximum level at the moment of ignition).

including the popular solenoid type or the more recent piezoelectric [71] ones. Consequently, the set I contains five independent variables. The number of variations for each independent variable is indicated in parenthesis.

1. P_{low}: low pressure on the injector inlet (5),
2. P_{high}: high pressure on the injector inlet (5),
3. dT_p: time interval of modulated pressure increase on the injector's inlet (5),
4. dT_v: time interval of the injector valve opening and closing (5),
5. T_{v1}: injector valve opening time (7).

The total number of variations of the independent variables is $5^4 \times 7$, which means 4,375 different sets of simulation boundary conditions.

Simulation output

For each combination of the independent variables the simulator computes three sets of time-dependent results. In other words, there are three families of function graphs in this data set. Furthermore, the following regular (scalar) dependent variables are computed. A list of the dependent variables is provided in Table 7.1.

Arbitrary shaped injection rate function graphs cannot be produced because of the physical requirements of the combustion in the engine. The engine will not run properly if the shape of the injection rate function graph does not approximately follow the ones in Figure 7.4. There

Function graph	Regular (scalar)
$Q_{inj}(t)$: injection rate	Q_p: amount of fuel injected during pilot injection
$P_{inj}(t)$: injection pressure	Q_m: amount of fuel injected during main injection
$A_n(t)$: needle lift	Q_{vo}: amount of fuel flowing back to the fuel tank
	V_{open}: needle opening velocity
	V_{close}: needle closing velocity
	L_p: spray penetration depth
	P_{ia}: average injection power

Table 7.1: Simulation results (dependent variables) of the fuel injection simulation.

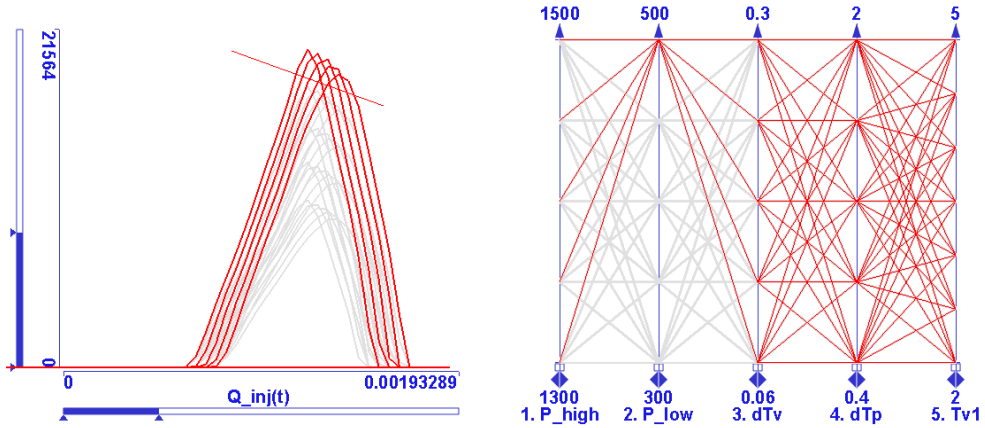


Figure 7.6: The function graph view is zoomed to show only the pilot injection. Pilot injections with high amount of fuel are brushed with a line brush. As seen in the parallel coordinates view of the independent variables, these all correspond to high P_{low} values.

are usually one (as in Figure 7.4) or two small peaks called *pilot injection* in the first quarter of the injection procedure in order to reduce combustion noise and NO_X emission in combination with the main injection. The goal is to find combinations of simulation parameters that control the volume of the pilot injection and produce the desired shape of the main injection.

7.1.3 Analysis of the Pilot Injection

We investigate how the amount of fuel and the timing of the pilot injection depend on the control parameters. We zoom to the pilot injection part of the $Q_{inj}(t)$ function graph and select function graphs with high peaks using a line brush (Figure 7.6). The corresponding items are highlighted in the parallel coordinates view of the input parameters. We can see that strong pilot injections are linked with high P_{low} values. We suspect that there is a direct correlation between P_{low} and the amount of fuel injected during pilot injection. To support this hypothesis we brush all P_{low} and high dT_v values with a color gradient brush (see Section 3.3.1) in a scatter plot view (Figure 7.7). The color gradient from red to green establishes visual links between the brushed items in the scatter plot view of the injection control parameters and the graph view of

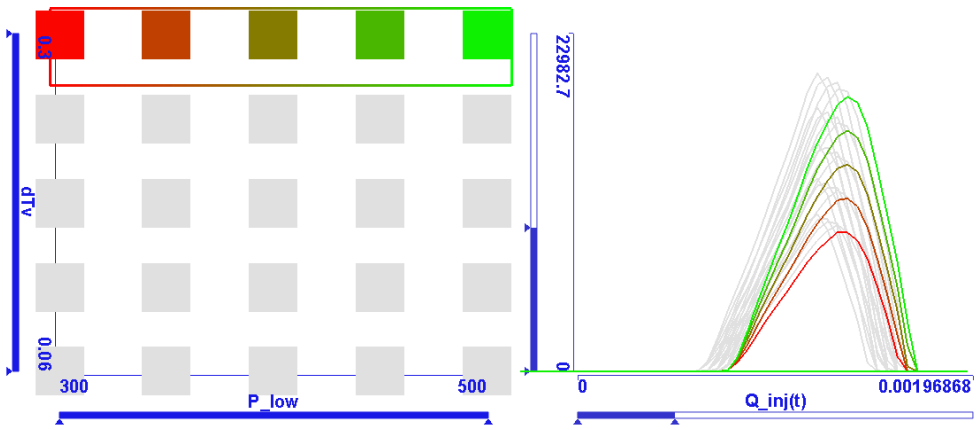


Figure 7.7: High dT_v and all P_{low} simulation parameters are brushed in the scatter plot diagram using a gradient brush. The correlation of P_{low} and the amount of pilot injection is revealed by the color gradient.

the injection rate.

Next, we try to find the parameters that determine the timing of the pilot injection. We brush the peaks of the function graph with a line brush and examine the parallel coordinates view of the control parameters. We conclude that the time of the pilot injection's peak depends on dT_v. This hypothesis can be counter-checked in an interactive way. A brush is moved in the scatter plot diagram of P_{low} and dT_v, and the highlighted injection rate function graphs are observed. We find that large dT_v values cause the pilot injection to start later and also to have slightly lower volume. The fully covered axes of the three other control parameters in the parallel coordinates view suggest that the pilot injection's shape does not depend on them.

7.1.4 Analysis of the Main Injection

The shape of the main injection is critical in achieving emission, efficiency, and combustion noise goals [219]. The optimal shape of the main injection is different for each particular engine operating point, shown in Figure 7.5. Note that this classification is somewhat arbitrary, since the shape changes from square to boot in a continuous manner as the control parameters vary. The engine control unit (ECU) measures engine speed and load to determine the current operating point. For each operating point the ECU contains a lookup table of injection control parameters used to control the injection system. The goal is to find suitable sets of control parameters for characteristic points in the diagram and understand how various properties of the injection rate function graph can be controlled. In the following we investigate how suitable control parameters can be found for specific main injection shapes. For each case we also demonstrate some additional dependencies and tendencies in the data set.

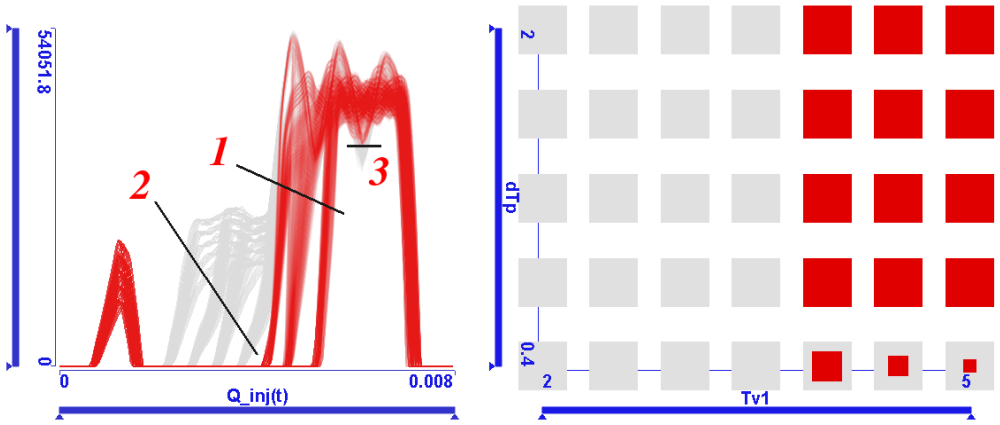


Figure 7.8: Ranges of control parameters that produce square shaped injections. First the user attempted to select square shaped graphs with brush 1. This brush selects several not square shaped graphs, too. These are removed by subtracting graphs that intersect brush 2. Brush 3 removes graphs that drop under a certain threshold in the main injection part. This property is a result of vibrations in the fuel line, which should be avoided.

Square shaped main injection

Square shaped main injection is desirable when load is very low or when the engine speed and load are both high. We used a combination of three line brushes to select square shaped injection rate function graphs (Figure 7.8). Brush 3 excludes undesired shock wave reflections.

Taking steps similar to the ones used when investigating the pilot injection we discover that T_{v1} is high for the brushed function graphs. That means the injector valve opens late when the pressure on its inlet is already very high. This leads to a sudden increase of injection rate, creating square shaped injection rate function graphs. As brush 3 is created we observe in the linked scatter plot diagram that most of the items with low dT_p (time interval of modulated pressure increase) are removed from the focus. This means dT_p must not be very low in order to avoid shock wave reflections.

We also study the desired needle opening and closing velocities and the correlations between the injection rate and the needle lift function graphs for this case. In order to do so, high T_{v1} and dT_p are brushed in the scatter plot diagram in Figure 7.9. The highlighted points in the V_{close}/V_{open} scatter plot show that fairly fast needle opening and closing is required for square shaped injections. The needle lift function graph (bottom right) is also linked. The color gradient of the brush shows a strong correlation between the needle lift and the injection rate graphs.

Ramp shaped main injection

Ramp shaped main injection is desirable when the engine speed and load are in a mid-range. In the previous case we have found a correlation between T_{v1} and the shape of the injection rate function graph. We also know that the time interval of the modulated pressure increase on the injector's inlet should be fairly high to avoid reflections. Based on this we start the investigation

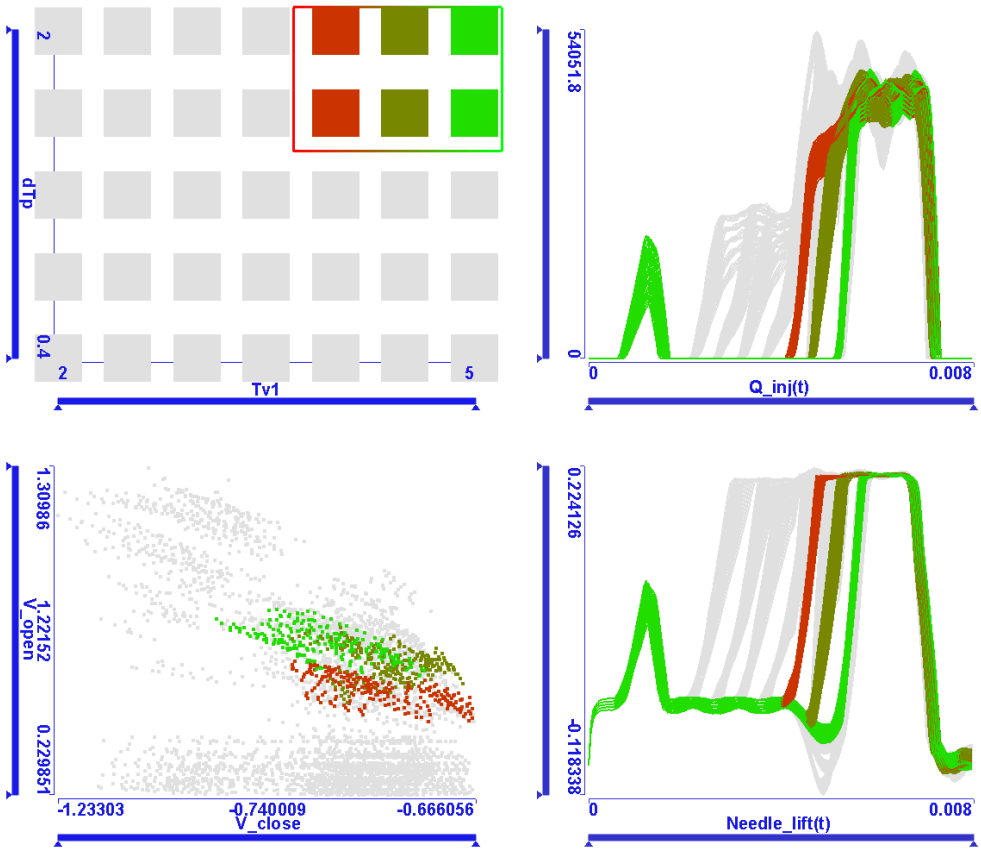


Figure 7.9: Required needle movement characteristics for square shaped main injections. Top left: control parameters that produce square shaped injections are brushed. Top right: red to green gradient shows that earlier valve opening times cause deviation from the ideal square shape. Bottom left: needle opening and closing velocities must be fairly high for this shape. Bottom right: the shape of the needle lift function graph is closely correlated to that of the injection rate function graph.

by brushing cases when the injector valve starts opening a little later, and we exclude low dT_p ranges (brush 1 in Figure 7.10).

The histogram of P_{high} is also brushed (brush 2). The intersection of the two brushes is studied in the injection rate and injection pressure function graphs. The corresponding function graphs of injection rate and pressure are similar in shape but differ in their maxima as P_{high} is varied (Figure 7.10).

Boot shaped main injection

Boot shaped main injection is desirable for engine operating points of mid-range engine speeds and high load. From our previous experience we assume that the injector valve has to be opened very early to achieve this shape. This assumption is verified in Figure 7.11 by brushing the

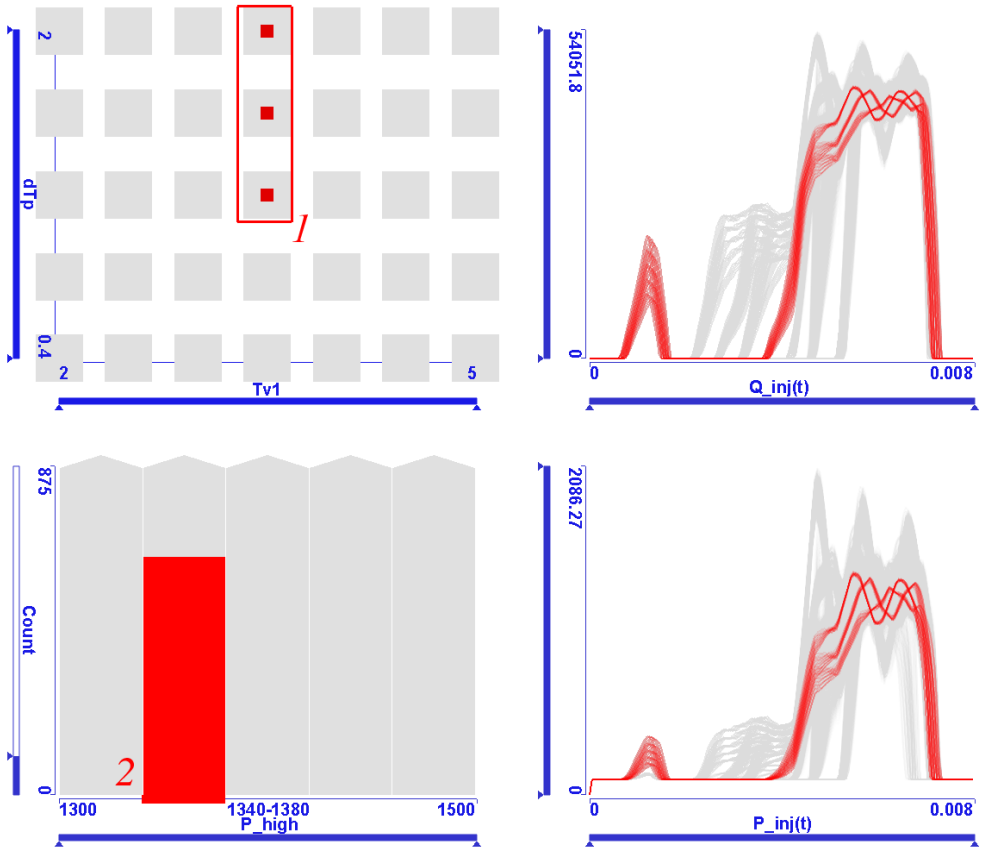


Figure 7.10: Top left: brush 1 selects control parameters for ramp shaped main injections. Bottom left: brush 2 selects different values of the high pressure on injector inlet. The two brushes are combined using the AND operation. Top right: injection rate function graphs of the brushed items. Bottom right: injection pressure function graphs of the brushed items. By dragging brush 2 and studying the linked function graph views we observe that injection rate function graphs have similar shapes but different maxima.

corresponding region in the scatter plot diagram of T_{v1} and dT_p.

Now we investigate how the desired amounts of fuel in the main injection and various injection penetration levels are achieved. The scatter plot diagram of these dependent variables is brushed and the brushed items are observed in the linked views in Figure 7.12. We observe that the brushed injection rate function graphs are all boot shaped. In the parallel coordinates view it is obvious that boot shaped injection does not require fast injection rate increase, but fast needle closing and injection rate decrease are still preferred. We also discover that for deep fuel spray penetration and high injection powers (brushed in green in Figure 7.12) fast needle closing velocities are required. The injected fuel mass (Q_m) and the amount of fuel returned to the fuel tank (Q_{vo}) are both fairly high. This matches our expectations, since we see in the parallel coordinates view that fuel pressure P_{high} was also quite high in these cases.

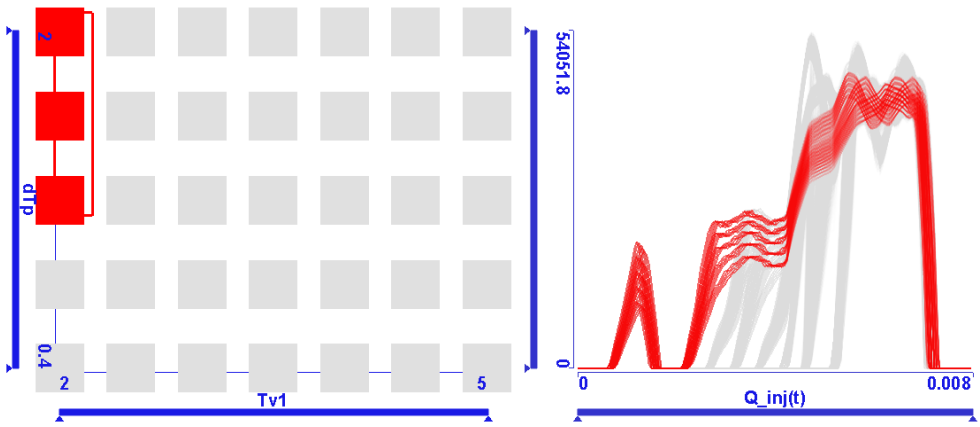


Figure 7.11: If the injector valve is opened very early then the injection rate quickly increases to the "boot" level. It reaches its maximum when the mixture is ignited in the combustion chamber.

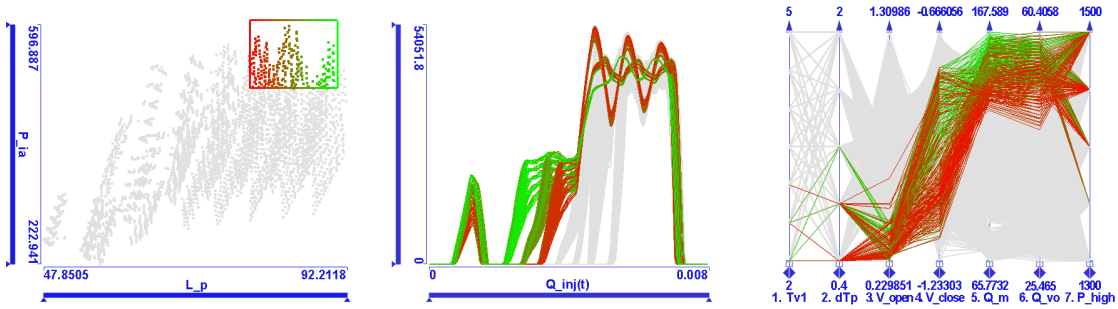


Figure 7.12: We investigate the conditions when fuel is injected deep in the combustion chamber and with high power. The corresponding items are brushed in the scatter plot diagram. The linked injection rate function graphs show that this requires boot shaped main injections. The desired needle opening and closing velocities are highlighted in the parallel coordinates view.

7.1.5 Insight Gained from the Analysis

In this example we have gained valuable insight into the fuel injection simulation data set and thereby into the fuel injection process. We found that the amount of injected fuel in the main injection stage can be controlled by adjusting P_{high}. The amount of pilot injection is controlled mostly by P_{low}, but dT_v also has some influence on it. We observed that choosing dT_p and T_{v1} is the key to achieving the desired injection shapes for various engine operating conditions. When the pressure on the injector's inlet increases too fast, then the resulting wave can be reflected into the fuel line, which impairs our control over the injection's shape. By studying the needle lift function graph and the related V_{open} and V_{close} simulation outputs we can define the desired needle characteristics for specific injection shapes and see how tightly the injection rate and the needle lift function graphs are correlated.

7.2 Interactive Visual Analysis of a Timing Chain Drive

The operation of the valves in an engine must be synchronized with the piston's motion. Most automotive engines use chain or toothed belt drives to transfer the rotation of the engine's crankshaft to the camshaft that operates the valves. Thus the camshaft rotation is synchronized with the piston's motion. Durability and low noise levels are two important goals when designing such timing drive mechanisms. Various factors such as high engine speeds, dynamic and inertial phenomena may cause undesired vibrations that increase noise levels and mechanical wear. This behavior can be simulated in software thus simulation tools are used extensively in the design of timing chain drives.

The design process of timing chain drives typically involves running computer simulations of many design variants with several combinations of input (control) parameter values in order to find an optimum. The analysis of many simulation runs poses new challenges. Previously, the analysis of those results at our industrial partner AVL-List GmbH (`www.avl.com`) was based on using static 2D diagrams [241] and the 3D visualizations described in Section 6.2 of this thesis. They are suitable for the detailed analysis of one or a few simulation variants, but they do not facilitate the interactive analysis of many simulation runs. In this section we demonstrate a case study of a timing chain drive optimization task using the segmented curve view introduced in Section 5.2, integrated into a coordinated multiple views system.

7.2.1 Simulation of Timing Chain Drives

Timing chains (or toothed belts) transfer the rotation of the engine's crankshaft to the camshaft which in turn actuates the valves, synchronized to the piston's motion. The crankshaft must complete precisely two rotations to one camshaft rotation. The chain's motion deviates from its ideal kinematic path especially at high engine speeds. Dynamic and inertial phenomena cause vibrations that increase noise levels and mechanical wear. The vibration also causes the accuracy of the motion coupling to deteriorate. Thus the camshaft's rotational velocity is not constant but contains undesired high frequency components. This can induce rougher and less controlled valve operation which can also reduce fuel economy, engine performance, and degrade emission quality. This behavior can be simulated in software thus simulation tools are used extensively in the design of timing chain drives.

Simulation model

Strictly speaking, timing drives consist of the chain or belt, sprockets on the crankshaft and on the camshaft(s), and one or more fixed or spring-loaded guides. The basic approach is to model each chain link as a single rigid body connected by spring/damper units to neighboring chain links, and simulate the resulting multibody system based on the Newton-Euler laws. The crankshaft and the camshaft sprockets are modeled as generic mass elements with only one degree of freedom (DOF), the rotation around the x-axis, which coincides with their natural axis. Similarly, guides can rotate around the x-axis on their pivot point. The motion of the chain is computed only in the y-z plane. The motion perpendicular to the plane can be ignored without

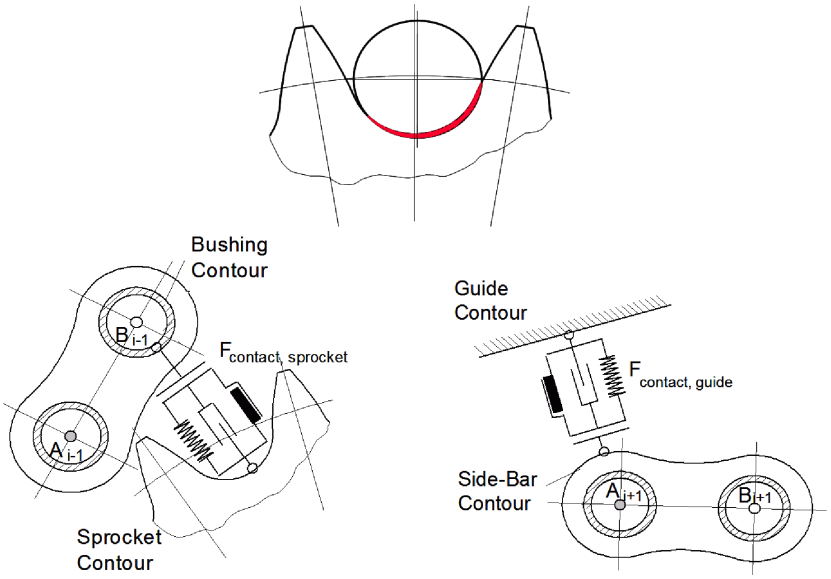


Figure 7.13: Contact forces between chain links and sprockets. The red region indicates the overlap area of the contact contours. Each chain link has smaller circular contours at both ends that model the bushing which comes into contact with the sprockets. The side-bar contour is the outer edge of the chain link that slides on guides.

influencing the validity of the simulation. Therefore, each chain link has three enabled DOFs: translation in y and z directions and rotation around x-axis.

In a simulation the objects are described using their contact contours. The contours of sprockets and guides reflect their actual shapes. The contours of chain links are a less intuitive. The contour of a chain link consists of circles around the pins that interconnect the chain links. There are two circular contours at each pin of the chain link. The smaller circle is the surface that comes into contact with sprockets. The larger circle is the outer surface of the link that slides on guides. Simulation of toothed belt drives works in a very similar way. The belt is represented as a spring-connected sequence of rigid sections, typically one section per tooth. The contour of each section is the cross-section of the respective part of the belt.

The simulation computes dynamic motion quantities of all chain links and forces between elements. There are two classes of forces: (a) contact forces between chain links and the sprockets/guides, and (b) connection forces between neighboring chain links. The contact forces act when a link comes into contact with a sprocket or guide. The algorithm for computing contact forces is based on evaluating the size of the overlap area between the contact contours of the two objects, the stiffness of materials, the relative velocities, and the damping properties of the materials. The connection forces act between neighboring chain links and are computed in a similar way. Figures 7.13 and 7.14 illustrate the corresponding connection and contact force models. For a more detailed explanation about the simulation of chain drives with rigid body dynamics please refer to the work of Sopouch et al. [241].

In this case study we use a model of a basic type chain drive system which consists of two

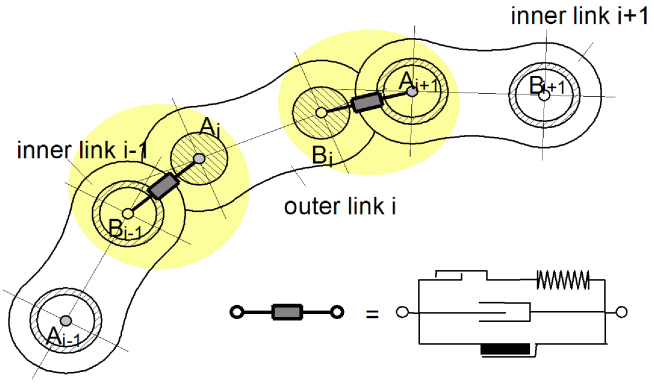


Figure 7.14: A model of the connection forces in the chain drive simulation. Each chain link is connected to its neighbors via stiff spring and damper elements.

Parameter	Values	Unit
sprocket stiffness	1.0E+7, 4.0E+7, 7.0E+7, 1.0E+8	[N/m]
guide stiffness	5.0E+6, 2.0E+7, 3.5E+7, 5.0E+7	[N/m]
chain preload	100, 200, 300, 400	[N]
sprocket offset	-0.5, 0.0, +0.5	[mm]
engine speed	1000, 2000, 3000, 4000, 5000, 6000	[rpm]

Table 7.2: Control parameters (independent variables) of the timing chain drive simulation.

sprockets, the camshaft sprocket (38 teeth) and the crankshaft sprocket (19 teeth). Two guides lead a bushing chain along the chain path to reduce lateral vibrations. The constant load is applied at the camshaft sprocket and a constant rotation is prescribed at the crankshaft sprocket. Lash in the chain and friction in the contacts between the chain and sprockets are not considered. A commercially available software package, EXCITE Timing Drive from AVL was used for the timing drive simulation. It is a multibody simulation software tool for the simulation of engine valve train components. The model is shown in Figure 7.15.

Simulation parameters

Several control parameters can be defined in the simulation software. In the scope of this case study we varied engine speed and four design parameters: stiffness of the sprocket material, stiffness of the guide material, chain preload and camshaft sprocket offset from the designed position in the z direction (see Figure 7.15). A positive offset means that the camshaft sprocket is further away from the crankshaft than its kinematically ideal position; therefore the chain becomes relatively short. Table 7.2 shows the ranges of variation of the parameters. With respect to the data model in Section 3.2, the control parameters are the independent variables in the data set. There are 1,152 possible combinations of these parameters or, in other words, 1,152 different simulation runs. In each simulation run the simulation period is long enough for all chain links to complete a full revolution.

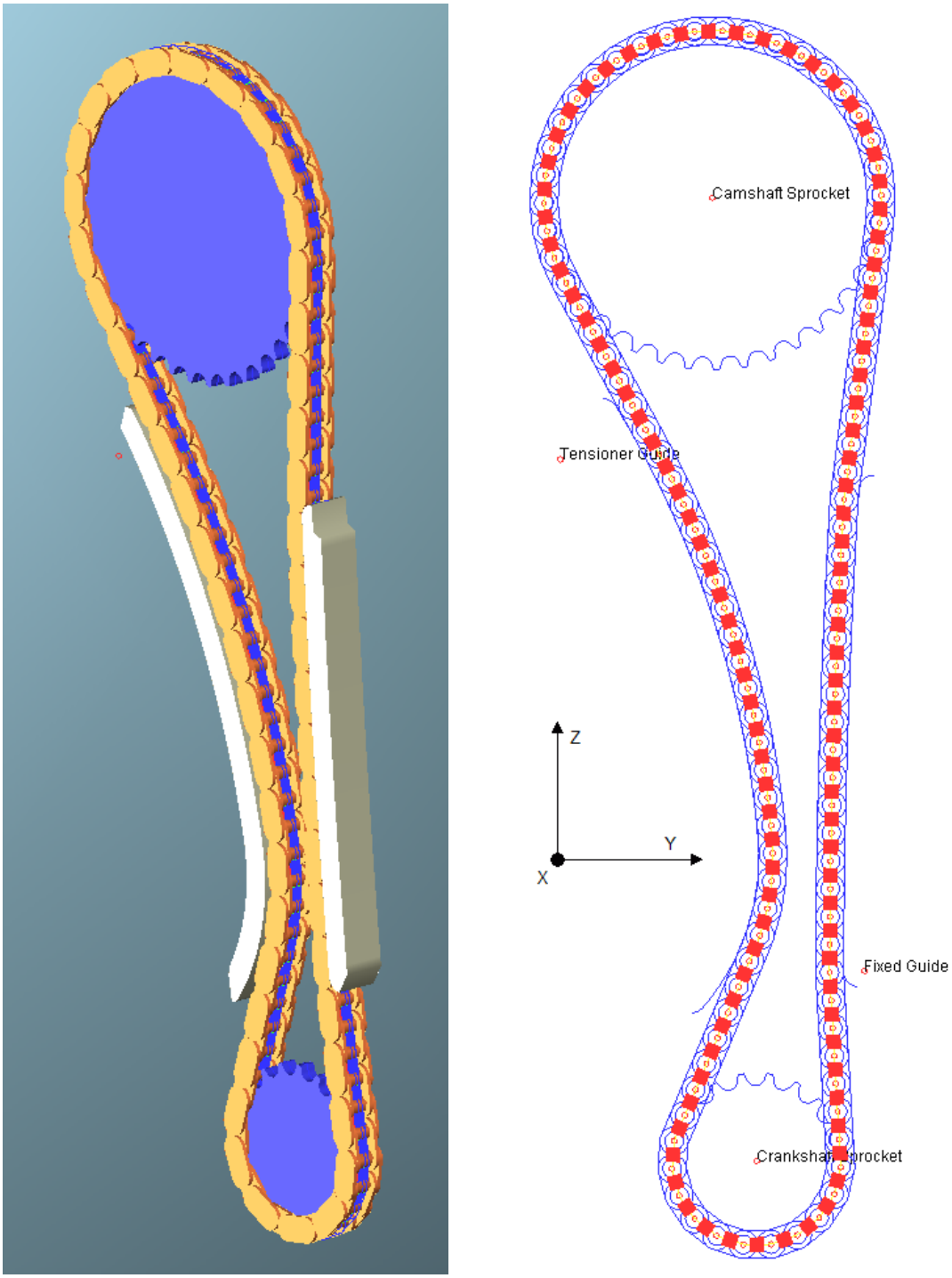


Figure 7.15: Left: 3D model of the chain. Right: the model simplified to simulation contact contours. Red boxes indicate the centers of gravity of chain links.

In the context of timing drive simulation, the simulation results are termed *response parameters*. In the visual analysis data model, they are the dependent variables. Three response parameters were computed for each simulation run from the raw simulation output described in Section 6.2.1. All three are function graphs.

- Maximum contact forces [N] over the simulation time for each chain link.
- Maximum connection forces [N] over the simulation time for each chain link.
- Fourier transform of the camshaft sprocket's rotational velocity [rad/s vs. orders].

The independent variable t in function graphs is often time. On the contrary, in the contact and connection force function graphs t rather represents a chain link numbered from 1 to 100, thus $1 \leq t \leq 100, t \in \mathbb{N}$. The value of the function graph represents the maximum of the contact or connection forces that act on the given link during the entire simulation time span. That implies that the function graphs are not continuous.

Engineers generally face four main tasks in the analysis of chain drive simulations:

- finding invalid parameter combinations
- parameter sensitivity analysis
- reducing chain noise
- keeping contact and connection forces within a range

In the following, we demonstrate many of the concepts introduced in the previous chapters, including iterative composite brushing, the segmented curve view, data aggregation, and the 3D visualization of rigid body systems are useful in the analysis of timing chain drive simulation data.

7.2.2 Finding Invalid Parameter Combinations

The first task engineers are interested in is checking if some of the control parameter combinations result in invalid responses. These combinations must be excluded from further analysis. This is an exploration process.

In order to check results, the curve view is used to investigate maximum contact forces. One clear outlier curve is visible in Figure 7.16. This graph shows that a number of chain links suffer extremely high contact forces.

We computed the maxima of the contact force and connection force function graph families (see Section 4.3.1), and plotted them in a scatter plot. The OR combination of two brushes selects large maximum contact or connection force values. In fact, a single point is selected in the extreme top right corner of the scatter plot. The parameters that lead to such system behavior are shown in the parallel coordinates view. This outlier comes from a simulation run where the very stiff camshaft sprocket is pushed 0.5 mm away from the crankshaft, the preload is small and the engine speed was high.

The 3D visualization (see Section 6.2) can be used to investigate this case in more detail. The animation reveals that the chain is vibrating wildly. Three frames of the animation are shown in Figure 7.17. We suspect that the aforementioned combination of parameters produces a resonance in the chain that generates extremely high forces. The behavior is actually so out of

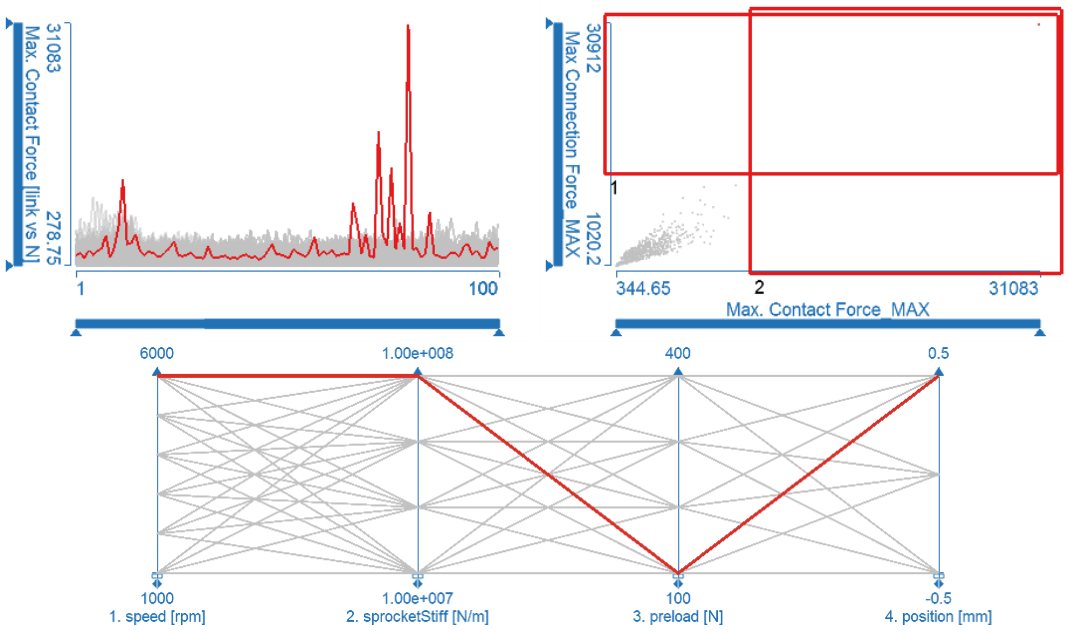


Figure 7.16: Finding invalid parameter combinations using three linked views. Top left: one of the contact force function graphs has a very unusual shape. Other graphs are shown in gray to provide context. Top right: the maxima of the contact force and connection force function graph families are plotted in a scatter plot. The OR combination of two brushes selects large maximum values. Bottom: the parallel coordinates reveal that extreme forces appear at 6000 rpm when the camshaft sprocket is offset by 0.5 mm, the sprocket material is very stiff, and the preload is small.

control that we assume that there may be some error in the simulation itself, too. Before running the simulation, the software computes the initial positions of bodies in the system. We suspect that this computation went wrong and failed to estimate the correct initial conditions for this case. This simulation run must be excluded from further analysis.

7.2.3 Parameter Sensitivity Analysis

Parameter sensitivity analysis is also an exploration process where the goal is to identify the main parameters and understand how changing those parameters influences the defined results [85, 255]. If one considers the simulation process as a black box, then this procedure can be called a black box reconstruction step (see Section 3.4.1).

First, we investigate connection forces at various engine speeds. The engine speed is not a design parameter that we can control but we want to understand how the results depend on engine speed. We brush various engine speeds and study the linked segmented curve view displays. Connection forces at 1000 rpm and at 4000 rpm are shown in Figure 7.18.

There are three clearly visible clusters of connection forces at 1000 rpm in Figure 7.18(a). We assume that there is a control parameter causing this clustering. This will be investigated

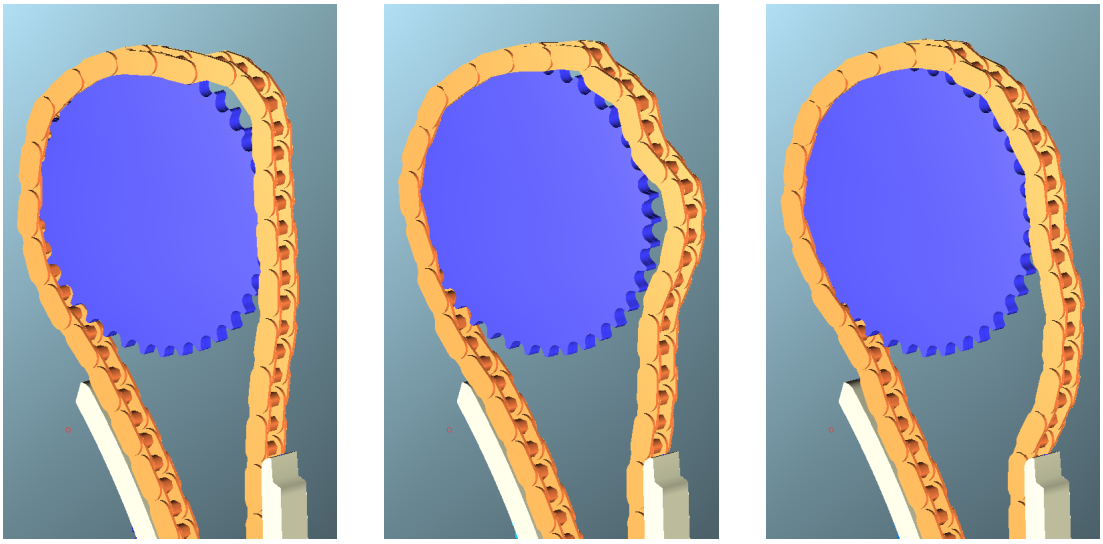


Figure 7.17: Three frames of the animated 3D view of the extreme chain vibration.

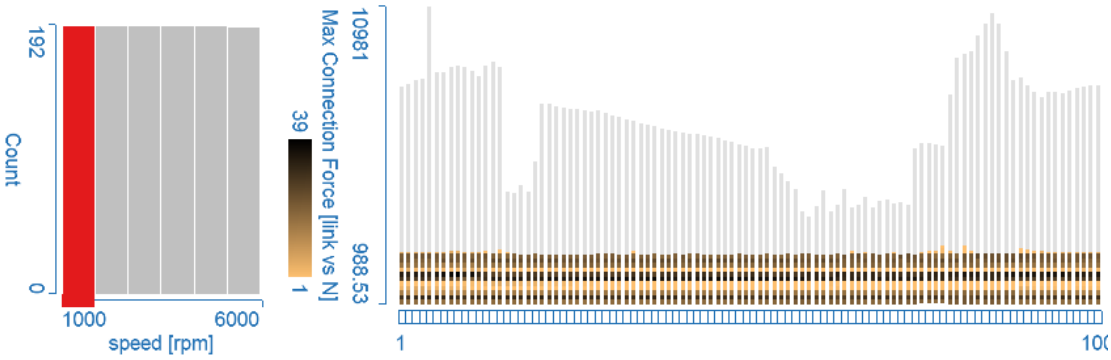
later. We also note that at 4000 rpm there are no clearly visible clusters. This indicates that the individual graphs exhibit more variety. We infer that it is probably more complicated to keep the connection forces in a specified range at higher engine speeds.

We examine the clusters noted in Figure 7.18(a). This time we brush 1000 rpm in the parallel coordinates view and then brush one of the three clusters of connection forces (Figure 7.19, brushes number 1 and 2). The corresponding sprocket offsets are highlighted in the parallel coordinates view. We can move brush 2 to select the other clusters and infer from the highlighted items in the parallel coordinates view that the low, middle, and high clusters correspond to -0.5 mm, 0 mm, and +0.5 mm sprocket offsets, respectively. Cases with high connection forces are not invalid, but they are not desired, since they cause increased wear.

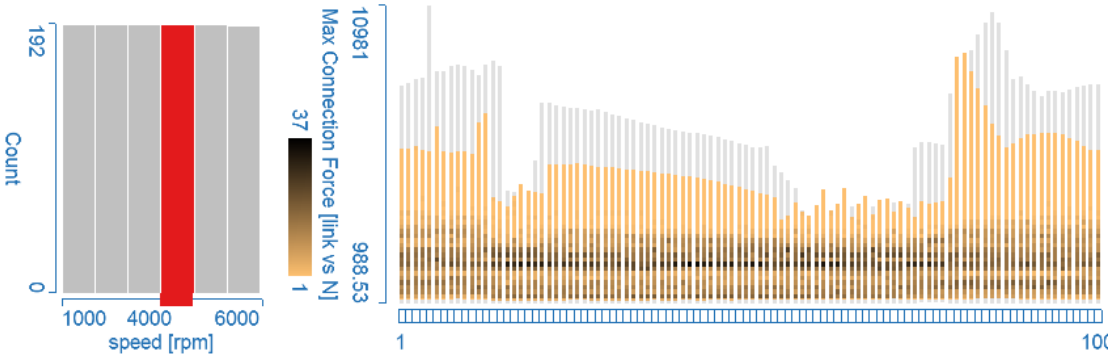
We can examine the influence of different preload parameter values by brushing the corresponding axis in the parallel coordinates view (brush number 3 in Figure 7.19). Larger preload values generate slightly larger connection forces, but the influence is far more subtle than that of the sprocket offset.

Finally, we examine the influences of the different guide and sprocket stiffness parameters on the contact forces. A snapshot of this interactive exploration is captured in Figure 7.20. We brush those two parameters in a scatter plot, move the brush to various combinations and study the contact forces. This would be a difficult task with the alpha-blended curve view. It is practically impossible to choose an alpha value which is transparent enough to reveal details in densely populated regions but at the same time opaque enough to preserve graphs in less crowded parts of the view.

The segmented curve view manages to display the details of the distribution in dense areas and the outlier graphs at the same time. We find that using stiffer material for sprockets increases the maximum contact force on chain links. Stiffer guide material has similar implications, but it also makes the distribution of contact forces less even. A very stiff guide results in some of the



(a) Connection forces at 1000 rpm.



(b) Connection forces at 4000 rpm.

Figure 7.18: Engine speeds of (a) 1000 rpm and (b) 4000 rpm are brushed and the corresponding connection forces are highlighted in the segmented curve view that uses 64 bins, global binning, and a relative color scale. The connection forces are rather varied at 4000 rpm, but at 1000 rpm there are three clearly visible clusters.

chain links suffering extremely high contact forces, while forces on others links is kept within a tighter range. It is obvious that this uneven distribution of forces causes extra wear of the more loaded chain links and it is more preferable to have an even distribution.

7.2.4 Optimization

In this section we investigate two optimization goals. We try to reduce chain noise and keep the magnitude and fluctuation of maximum forces low in order to reduce dynamic load on the chain that would cause extreme wear.

The level of chain noise is in correlation with the magnitudes of the contact forces. The noise spectrum is related to the higher order components of the camshaft sprocket's rotational velocity. Therefore, we want to minimize peaks in the Fourier transform of the rotational velocity. We first examine what conditions lead to high peaks in the spectrum. Figure 7.21 shows that these peaks appear at higher engine speeds only. However, not all of the histogram bars corresponding to

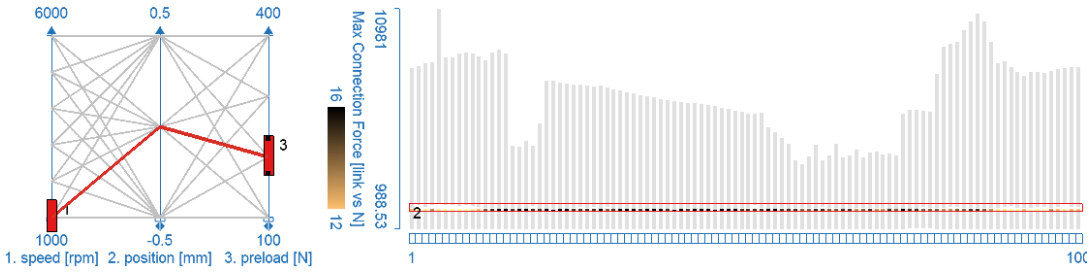


Figure 7.19: First, engine speed of 1000 rpm was brushed in the parallel coordinates view. This highlighted three clusters in the segmented curve view of connection forces (Figure 7.18(a)). An AND brush (see Section 5.2.3) spanning all bars was used to select the middle cluster. The corresponding sprocket offset is shown in the parallel coordinates view. Brush number 3 narrows the focus to a specific preload setting in order to examine the additional influence of preload on the forces. By moving brush 3 we discover that higher preload causes slightly higher connection forces.

high engine speeds are highlighted. A part of even the 6000 rpm bar remains gray, which means there are parameter combinations which avoid those high peaks in the spectrum at 6000 rpm.

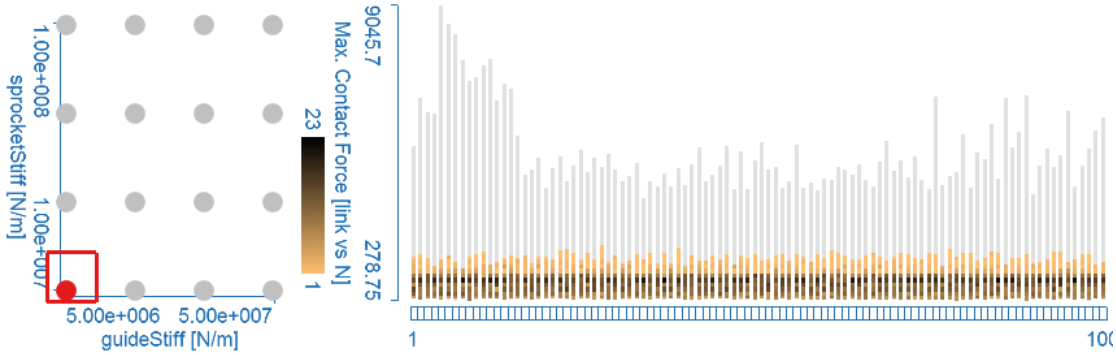
There is an interesting peak at the 19th order (located at the right end of the spectrum). This is related to the "polygon effect" specific to chain drives [225]. Chain links engage and disengage to/from the sprocket as it rotates. That produces a whining noise with a frequency of $n * (u/60)$ where n is the number of crankshaft sprocket teeth and u is the engine speed (rotations per minute).

There is another important consequence. The sprocket's moment is distributed onto a different number of chain links depending on the sprocket's angular position. This change in the number of engaged chain links can be considered as a discrete process due to the high contact stiffness. These abrupt changes in load lead to oscillations in the sprocket's rotational velocity and in the chain connection forces with an order that equals the number of crankshaft sprocket teeth. This peak has been brushed in the segmented curve view in Figure 7.22 and we discover that all simulation cases at 1000 rpm are highlighted in the histogram. This indicates that the polygon effect cannot be completely eliminated with any combination of parameters used here. It can be minimized, though.

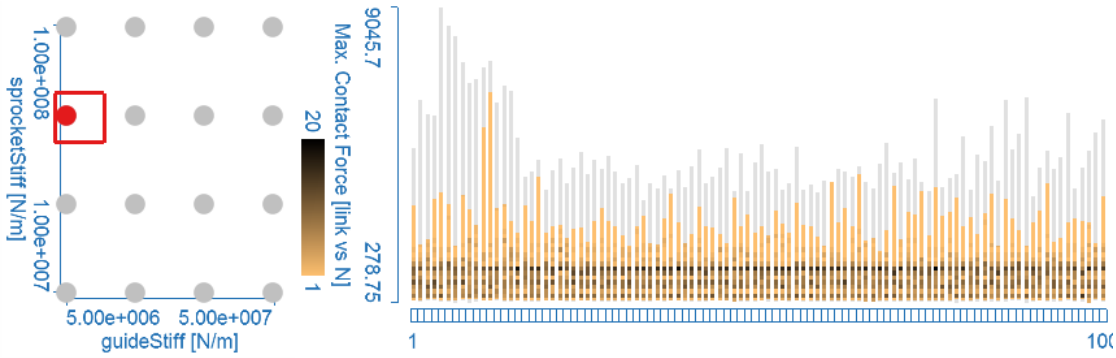
We now try to find the parameter values that produce optimal chain behavior. The ideal case has low contact and connection forces and the fluctuation of forces should also be minimal. The camshaft sprocket's rotational velocity spectrum should not have high amplitude components, especially at higher orders, in order to reduce high frequency noise.

We now have an overview of how the parameters influence the result. We want to narrow the set of design parameter values progressively to a single combination which provides the best compromise. This is a highly interactive process where the analyst creates and moves brushes and examines the linked views of the response parameters.

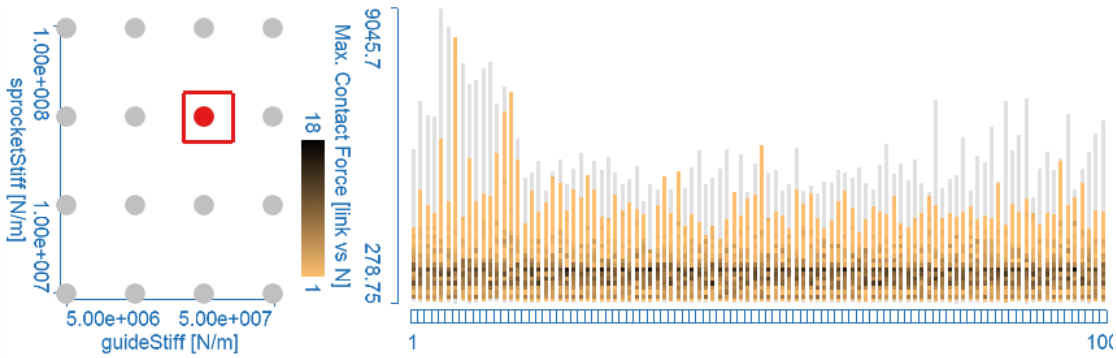
We brush different ranges of control parameters in a parallel coordinates view of the design parameters and observe the linked function graph views of connection force and contact force diagrams as well as a segmented curve view of the spectrum of the sprocket's rotational velocity



(a) Low guide and sprocket stiffness produces low contact forces. The distribution is very even, which means similar forces act on all chain links.



(b) Higher sprocket stiffness creates higher contact forces. Very few chain links have exceptionally high forces. The reason for the three-way clustering is again the different sprocket offset parameter. Compare to Figure 7.18(a).



(c) Stiffer guide material also increases the contact forces, and it makes their distribution more varied, too. Some chain links suffer considerably larger contact forces than others.

Figure 7.20: Three snapshots showing the influences of different stiffness parameter values on contact force. The guide and sprocket stiffness are displayed in the scatter plot, and a brush is moved to select different combinations. The contact forces are displayed in the linked segmented curve view.

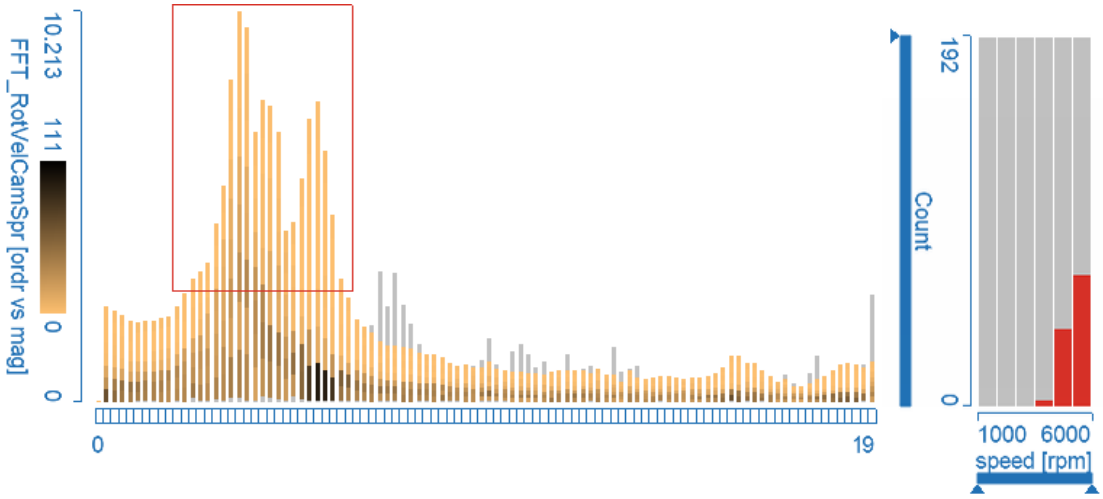


Figure 7.21: High peaks are brushed with the red rectangle in the Fourier transform of the camshaft sprocket's rotational velocity. This OR brush (see Section 5.2.3) selects all graphs that enter the rectangle. The histogram shows that these peaks appear at high engine speeds. Note that in this segmented curve view we use local binning (see Section 5.2.1) because it is more useful for the visualization of the wide variations in the maxima.

to see its distribution. Figure 7.23 captures the final snapshot of this process.

Figure 7.19 shows that sprocket offset settings above -0.5 mm produce undesirably high connection forces so we brush -0.5 mm sprocket offset. Different preload parameter values are then brushed and the logical AND of the two brushes is shown in the linked views of the response parameters. The preload of 400 N is selected because the spectrum of the rotational velocity shows the smallest peaks in the segmented curve view, while the maximum contact and connection forces are kept relatively low.

Finding the optimal stiffness of sprockets and guides is the next target. We achieve that by adding new brushes to the parallel coordinates for those parameters and moving them while observing the other three linked views. The smallest sprocket stiffness creates the smoothest contact force diagrams, as expected. However, while the lowest guide stiffness also produces low forces, the spectrum of the rotational velocity has a more expressed peak at the 7th order. The guide stiffness value selected in Figure 7.23 is slightly larger, because that reduces the 7th order amplitude. The connection and contact forces are still relatively low and with little fluctuation which is desirable. Unfortunately, the peak at the 19th order in the spectrum is still there but its amplitude is reduced. As we can see in Figure 7.22 we cannot remove this peak completely with this range of design parameters.

Considering all these factors, we find the optimal set of design parameters that provides the best compromise for noise, motion coupling accuracy, and durability. Figure 7.23 shows the optimal parameters and the corresponding simulation response. Note that we use the alpha-blended curve view to display the contact and connection forces in this figure. The reason is that we are interested in the actual exact shapes of those graphs rather than their distributions since

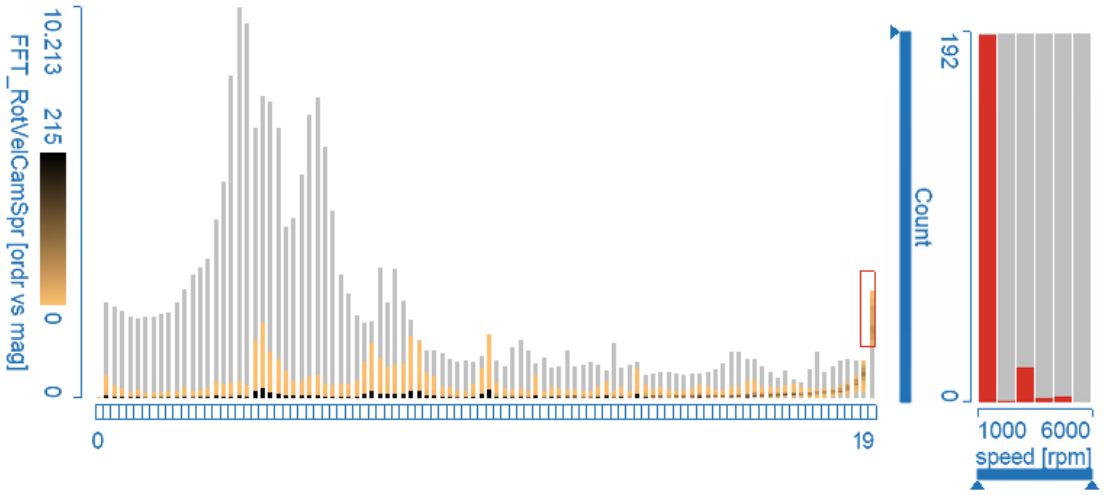


Figure 7.22: We brush the peak at the 19th order of the FFT of the sprocket's rotational velocity (red rectangle at the right end of the segmented curve view). That peak is related to the polygon effect specific to chain drives. All simulation cases of 1000 rpm engine speed are highlighted in the histogram. That means the polygon effect at 1000 rpm cannot be removed by any combination of the parameters.

we narrowed down the focus to the optimal set. The alpha-blended curve view works better for this purpose than the segmented curve view. A summary of the optimal design parameters is given in Table 7.3.

Parameter	Value	Unit
Sprocket stiffness	1.0E+7	[N/m]
Guide stiffness	2.0E+7	[N/m]
Chain preload	400	[N]
Sprocket offset	-0.5	[mm]

Table 7.3: Optimum parameters of the timing chain.

7.2.5 Insight Gained from the Analysis

We have analyzed a timing chain drive simulation data set in this section. Different sprocket and guide stiffness, sprocket offset, and chain preload parameters were used in a total of 1,152 simulation runs and the chain's motion was computed for each combination of the parameters and at different engine speeds. During the analysis, we identified an invalid parameter combination which produced extremely high contact forces. We explored how connection and contact forces depend on the parameters. We learned that the connection forces between chain links depend primarily on the sprocket's offset and, to a lesser extent, on the chain's preload. The contact forces depend on the sprocket and guide stiffness settings. We also found that if the guide ma-

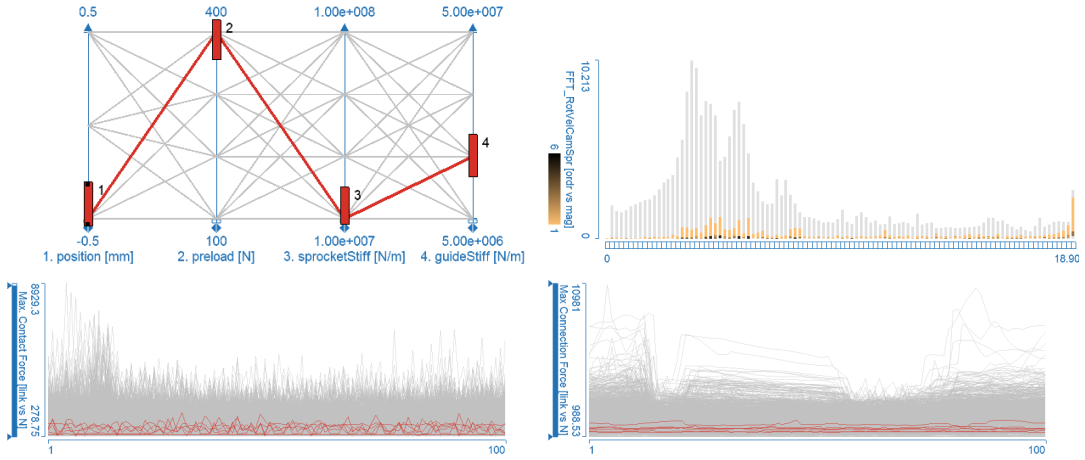


Figure 7.23: This snapshot captures the last iteration of the optimization process and shows the optimum design parameters. Top left: The design parameters are brushed in the parallel coordinates view. Top right: The segmented curve view shows that the Fourier transform of the sprocket's rotational velocity has few peaks and they have small amplitude. Bottom left: Maximum contact forces are small at all chain links. Bottom right: All chain links have very small maximum connection forces.

terial is very stiff, then some chain links suffer significantly larger contact forces than others. After understanding the dependencies between parameters and forces, in an iterative optimization process we could find a combination of parameters that represents a favorable compromise in reducing noise and maximizing durability.

Chapter 8

Summary

"The existing scientific concepts cover always only a very limited part of reality, and the other part that has not yet been understood is infinite."

— Werner Heisenberg (1901–1976)[1]

Computational simulation is increasingly used to assess the quality and potential of new designs in the automotive industry [188, 225], as well as in other application domains [17, 65]. The currently available computational power makes it possible to compute many repeated simulations of the same model with different input parameters, representing different operating conditions and design choices. The analysis of data sets produced by such multi-run [118] simulations is especially challenging, because the data are relatively large, often time-dependent, and high-dimensional; both in terms of independent variables, i.e. simulation parameters, and dependent variables, i.e. result attributes. Computational analysis methods are widely used, but they are not always sufficient, for the analysis of such data sets; especially when trends, patterns, relations, and dependencies are explored.

In this thesis we introduce new methods for the interactive visual analysis of automotive engineering simulation data sets. Section 8.1 describes a novel data model for time-dependent, multivariate simulation data. We introduce a coordinated multiple views framework and novel visualization techniques for such data sets. The framework provides interaction features that support interactive analysis. In Section 8.2, we discuss the computation of derived data attributes. Tight integration of the computation of derived attributes into the analysis workflow is necessary when the system needs to support complex analysis goals that cannot be accomplished by traditional visual analysis methods of the original data set.

Simulation data often has a relevant spatial context, which calls for (generally domain-specific) spatio-temporal visualizations. In Section 8.3, we introduce glyph-based visualizations for rigid and elastic multibody systems. Section 8.4 contains a summary of two case studies that were done with engineers using our system, as well as other application examples outside engineering.

[1] German theoretical physicist, received the 1932 Nobel Prize in Physics for the creation of quantum mechanics.

8.1 Interactive Visual Analysis of Families of Function Graphs

In this section we introduce a data model and a framework of novel visualization and interaction techniques that support interactive visual analysis of high dimensional and time-dependent data. We describe common analysis procedures and exemplify how our system supports them.

Data Model

Multi-run, multivariate, and time-dependent simulation data can be effectively represented by a data model consisting of m *independent variables* (the simulation parameters) and n *dependent variables* (simulation results). The independent variables have scalar values and can be expressed as $x = [x_1, \ldots, x_m] \in I$. Here I denotes the set of all possible combinations of values of independent variables, representing all simulation runs. Dependent variables are functions of the independent variables and they constitute a subset D of the data set. There are two types of dependent variables, regular and function graphs, the latter representing time-dependent results. When the value of a dependent variable k is a function graph (data series over t) $f_k(x, t)$, it contains o data elements, one for each value of $t \in t_1, \ldots, t_o$. A *family of function graphs* is then a set of function graphs for each possible value of x, $f_k(x_i, t)|\forall x_i \in I$. In other words, a family of function graphs represents the time-dependent results of the same physical attribute from all simulation runs. Considering function graphs an atomic type in the data model opens new analysis opportunities.

Visualization Framework

The visualization framework offers a combination of *linked views* including standard histograms, scatter plots, and parallel coordinates for scalar attributes, as well as specialized views for families of function graphs. The user can create and arrange views as desired, and can have more than one instance of the same view type showing any combination of independent and dependent variables. Views can be temporarily maximized to facilitate detailed examinations. The entire analysis session (arrangement of views, data, and brushes) can be saved. Exchanging those session files provides a means of offline collaboration among analysts.

A set of conceptually simple, easily combined, and intuitive *visual operators* (brushing techniques) support analysis and exploration. The visual operators are used to select a subset of "interesting" values for the specific variables in *any* of the views. The selection is immediately highlighted in all other views. All views provide consistent *focus+context* visualization. There is a linked tabular view of raw data values that facilitates quantitative assessments. We have applied an optional *color gradient* along the brush and used this color gradient in the linked views to establish a visual link between the correlated data items. The color gradient also helps in discovering tendencies in the data set.

Complex queries can be formulated using *iterative composite brushing*. Several brushes can be defined in the same or in different views, and they can be combined by logical operations. We did not develop a complex *feature definition language* [59], but realized composite brushing by AND, OR, and SUB operations where the first operand is always the result of the latest composition. This results in a simplified, intuitive, and iterative workflow. Every new brush alters the

current selection status according to the composition rule in use. The user can interactively refine (using AND and SUB) or broaden (using OR) the current selection and steer the information drill-down. The user can also resize or move any existing brush in the chain. In conventional views, the user can select histogram bins, rectangular areas in scatter plots, or ranges of parallel coordinates axes [158]. Brushing function graphs is discussed in the following.

Visualizing Families of Function Graphs

We offer three visualization techniques for families of function graphs: the function graph view, the segmented curve view, and the color lines view.

The *function graph view* is essentially a line chart that displays all function graphs of a family at once. In order to reduce visual clutter produced by overplotting, we can render the pixels where more graphs pass through with higher intensity, effectively producing an alpha-blended overlay of line charts. The function graph view offers two brushing methods. The *rectangular brush* selects all function graphs that pass through the rectangle, similar to the timebox widget [96]. The brush can be limited to function graphs that enter and leave the brush at given edges. This facilitates the selection of function graphs that have a local extrema in the rectangle. The *line brush* is a simple line segment drawn in the function graph view. It selects all function graphs that it intersects. We have found compositions of line brushes very intuitive and effective in selecting or excluding function graphs whose approximate shape is known.

If the data series are not continuous, then the continuous lines in the function graph view falsely suggest continuity. Furthermore, choosing a transparency factor that reveals the distribution in densely populated regions and preserves outliers at the same time is not always possible. We address those issues with the novel *segmented curve view*. In that view, bars extend from the minimum to the maximum values across the family of function graphs for each fixed value of the independent variable of the functions. The bars provide segmentation along the independent axis. Each bar is divided into segments (bins) with a color that represents the number of function graphs passing through that segment. We offer different binning strategies to make the direct comparison of distributions in different bars possible, and to provide finer resolution in bars whose minimum and maximum values are closer.

The segmented curve view provides focus+context visualization (see Figure 5.6 on page 58). The bins containing one function graph are filled with the first color of the color gradient. The last color of the color gradient is used for the most populated bin of all. Alternatively, it can be used for the most populated bin in the focus set, in order to retain better resolution in the focus.

We offer a rectangular brush that can span multiple bars. There are two ways to interpret a brush that spans multiple bars: (1) a function graph can be selected if it enters and leaves the rectangle through its vertical edges, or (2) a function graph can be selected if it enters and leaves the rectangle through any edge.

Finding clusters in families of function graphs is a common task. We propose a pixel-based technique, the *color lines view* to support such tasks. Each function graph is displayed as a row of pixels. The horizontal positions correspond to values of the independent variable of the function. The corresponding function value is indicated by the color of the pixel. The rows representing individual function graphs are placed directly above one another to display an entire family. The color lines view also offers focus+context visualization (see Figure 5.11 on

page 65). The arrangement of rows is important in the discovery of clusters and correlations. We offer three criteria to control the arrangement: (1) sorting based on the values at a given horizontal coordinate, (2) sorting based on the value of another variable, and (3) brushing. Brushing does not change the order of rows, but marks a range of rows to be observed in subsequent permutations. Combinations of these simple operations can be used to solve tasks like finding patterns in a family of function graphs or understanding inter-dimensional correlations.

Analysis Procedures

From the visual analytics point of view, the goal of the analyst is to discover, in an iterative manner, trends, patterns, clusters, tendencies, and outliers in the data and to see how patterns in D map to the corresponding subsets in I and vice-versa. In the following we describe some typical analysis task in more detail.

Computational simulation of complex physical systems can be considered a "black box" that returns output for an input parameter set. The parameters can have so diverse effects on the results, that it is more feasible to reconstruct the black box by exploration rather than by trying to deduce its internals from a simulation process. To *reconstruct the black box*, the analyst fixes values of some independent variables to reduce the focus area and varies other independent variables while studying the corresponding dependent variables and function graphs. This is an interactive and iterative data exploration process: brushes are created and moved to areas of interest. When we have built up an overview of the dependencies we want to zoom in on details in both I and D in order to discover more subtle correlations in the data.

Finding combinations of independent variables that produce the *desired shapes of function graphs* is a common task. The procedure requires focus+context views of the graphs where criteria can be defined to select graphs of specific shapes, for example, by using combinations of line brushes. The related values of independent variables are highlighted in the linked views.

When finding *correlation between families of function graphs*, we want to investigate features of one family of function graphs depending on the properties of a set of function graphs in another family. This requires the simultaneous display of the families. The user brushes groups of function graphs in one family, and studies the corresponding ones in the other families. We may also want to narrow the search by filtering on the function graph's independent variables $\mathbf{x}$.

There is a strong need in engineering applications to perform automatic optimization of designs using several simulation iterations with suitably varied boundary conditions. The automatic optimization process must have an approximate model of the simulation in order to know how boundary conditions should be adjusted in search for an optimum. Visual analysis can be used to *create hypotheses* and rules that the automatic optimization can use in its simplified model and also to find out if the optimization misses some families of function graphs while searching for an optimum.

8.2 Analysis using Data Aggregation and Derivation

We propose a three-level classification of complexity in visual analysis tasks. Levels one and two represent the current state of the art. The third level requires advanced interaction and data

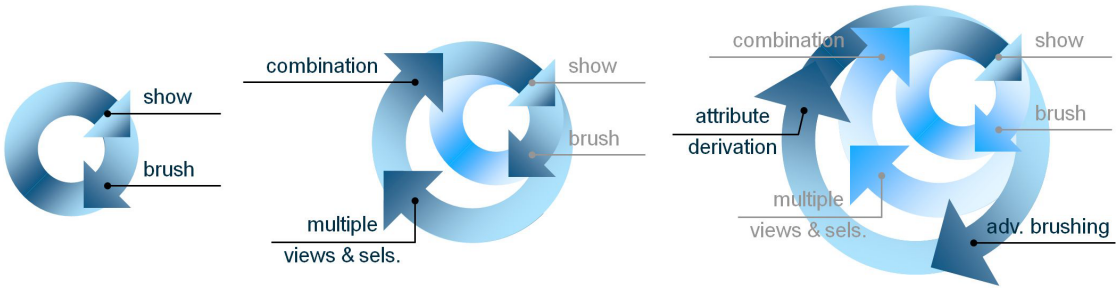


Figure 8.1: Interactive visual analysis on three levels of complexity. The first level is linking and brushing with one brush. On the second level, more views are used and brushes are combined with logical operators. On a third level, advanced interaction (brushing) and attribute derivation are added.

manipulation features so that deeply hidden implicit information from complex data sets can be extracted. Based on our experience gained from several case studies, we present some typical, complex analysis tasks of data sets containing families of function graphs and discuss how the visual analysis framework can support them.

Three Levels of Complexity in Interactive Visual Analysis

Interactive visual analysis is an iterative process. It usually starts with a simple analysis of the original data to gain overview. Then, for a more advanced analysis, the analyst needs to use more complex procedures and can combine findings from earlier stages of the analysis. The need for a more advanced analysis arises as information should be extracted which is more difficult to access. At a certain point it becomes difficult or even impossible to find features of interest by analyzing the original data only using conventional IVA methods. More advanced and complex interaction possibilities are required. Alternatively, or perhaps in addition to that, the data can be enhanced by computing aggregates or first derivatives of time series, for instance. Figure 8.1 illustrates our view on the three levels of complexity in interactive visual analysis.

In a coordinated multiple views system, the *first level* (smallest circle, left in Figure 8.1) can be expressed as simple linking and brushing with one brush. The user interactively selects some items in one view, and they are highlighted consistently in all views. Then a different set of items (another *feature of interest*) is brushed and the highlighted patterns in the linked views are studied. The user repeats this process, engaging in an iterative IVA loop. This is sufficient in many cases, but we cannot formulate more complex queries, spanning several attributes.

On the *second level* (larger circle in the middle of Figure 8.1) more views are used and several brushes can be combined with logical operators. Queries can be expressed as combinations of criteria on several (original) data attributes. However, this is insufficient to achieve many analysis goals, for example, selecting those function graphs in a family that are increasing at a given value of the independent variable; or finding curves of some specific shape.

In order to select increasing function graphs, an advanced brush can be designed, which considers the slopes of the lines. Another possibility is to compute the first derivatives of the

function graphs, and brush high values of the first derivative to select increasing function graphs. The combination of complex interaction and on-demand computation of such synthetic, *derived attributes* constitutes the *third level* (rightmost circle in Figure 8.1) of visual analysis, supporting complex analysis goals. Note that each of the three circles in Figure 8.1 encloses the previous ones from the lower level(s), indicating that a seamless transition between circles (levels of analysis), and also their integration, is possible.

The advantages and drawbacks of the two essential building blocks of complex analysis, advanced interaction and attribute derivation, are complementary. *Advanced interaction* does not increase the amount of data, nor does it usually require additional views, but the user needs to mentally manage the advanced interaction method. New types of analysis tasks may require new, specialized interaction techniques.

Attribute derivation increases the amount of data and usually requires additional views. The synthetic data attributes are often exactly the ones that are used in computational analysis, thus experienced analysts are familiar with them. Well-known, simple mechanisms can be used to interact with the data. Due to the step-by-step approach, with a sufficiently rich set of basis operations, a lot of very different derivations are possible, also ones that were not necessarily anticipated at the time when the IVA system is designed.

In open and flexible IVA systems that can be used for a variety of problems, attribute derivation may be preferred. However, in more targeted IVA solutions, where similar problems need to be solved repeatedly, advanced interaction can be more time-efficient for the experienced user.

An interesting analogy can be drawn between the three levels of complexity in analysis and the three thresholds of response times in human-computer interaction (refer to Card et al. [41]). The users' actions in level one analysis are spontaneous, and instantaneous response is expected from the computer. At level two, the users' actions are more complex, the pace of action becomes slower, and response within a second is sufficient to avoid the interruption of the users' cognitive process. At analysis level three, the users' decision to request the computation of new, synthetic data attributes reflects elaborate thinking. They are likely to accept longer response times, provided they have some feedback in the form of a progress bar or the visualization being progressively updated.

Complex Analysis of Families of Function Graphs

Based on our previous case studies, we identified some typical, complex analysis tasks associated with data sets containing families of function graphs. We demonstrate that they can be supported by specialized, advanced interaction techniques, and also by the computation of derived data attributes. Our work is in part motivated by curve sketching, where new attributes (e.g., extrema) and curves (e.g., first derivative) are computed to analyze single curves. We propose related analysis methods for entire families of function graphs. The proposed ideas are in part suitable for the analysis of families of generic curves, too.

Some of the typical goals in the analysis of families of function graphs can be effectively tackled by computing some *aggregate* (e.g., minimum, maximum, arithmetic mean, percentile, and integral) of each member curve of the family. A family of function graphs is then represented by a set of scalars, significantly reducing data complexity. Thereby, standard visual analysis tools

(e.g., histograms) available for scalar data can be used in the exploration of families of function graphs.

In many fields, a typical task when analyzing families of function graphs is finding curves that have a specific *slope* at a certain point in time. This can be achieved if the user can trigger the computation of the first derivative of a family of function graphs. A new function graph dimension is created and displayed in a new view, where items can be selected, for example, using a line brush. Once the first derivative is computed it can be used as input to aggregation. When looking for function graphs which rise steeply, the first derivative can be computed first, and then its maximum scalar aggregate can be brushed in a histogram. We can also select function graphs having a certain slope by enhancing the interaction. The *angular line brush* [161] selects function graphs that intersect the line at a specific slope interval.

Engineers often look for function graphs of certain *shapes*. In simple cases, combinations of line brushes are sufficient to select the shape. In general, this problem, too, can be solved using more advanced interaction, or by deriving attributes and using simple brushes. The similarity brush [35, 96, 178] represents the solution using advanced interaction: the user can either sketch a curve, or pick one of the curves; and all similar curves are selected. Alternatively, a set of specific scalars describing characteristic features of the curves can be computed and the desired values of those aggregates can be brushed [166].

Finding *correlations across families* of function graphs is generally accomplished by displaying the two families in two separate views and using the CMV system to compare them. We propose using phase diagrams to visualize two families of function graphs with a common independent variable (e.g., time) in a single view, in order to support such analysis. The phase diagram shows curves defined by the corresponding pairs of function graph values from the two families. This plot often reveals interesting relations between different dimensions of the data set that would be more difficult to discern using only function graph views. The phase diagram is also fully interactive. Like the function graph view, it supports angular line brush and similarity brush which makes finding and brushing hidden relations possible.

8.3 Interactive 3D Visualization of Multibody Dynamics

Rigid and elastic multibody systems are commonly used to model and simulate certain components of internal combustion engines. We introduce an interactive, glyph-based framework for the visualization and visual analysis of such multibody dynamics. The framework consists of an arbitrary number of 3D visualizations showing the model from different viewpoints. The views can be animated in concert.

It is possible to jointly analyze rigid and elastic body simulations. This is very relevant, because the components of the same engine are simulated using different principles. Timing drive components are modeled as rigid bodies, while cranktrain components use elastic body simulation. The two simulation models can be coupled during computation to represent the interaction between engine components. It is very important that the resulting multi-model [118] data set can be jointly visualized and analyzed in the same 3D view.

Rigid Multibody Dynamics

In the automotive industry, *rigid body simulation* is extensively used to compute the motion of timing drive components, including timing chain drives. Each link in the chain, as well as the sprockets, idlers, and guides are represented as rigid bodies. The simulation model uses a simplified set of *contours* to represent the bodies, instead of actual 3D models. The simulation computes the motion of bodies over several thousands of time steps, representing several complete revolutions of the engine.

In our proposed *glyph-based visualization*, bodies are displayed using their actual position and orientation, with icons indicating the contours of the body, the location of the center of gravity, and the local coordinate system. Additional simulation results, such as translational and angular velocity and acceleration, forces, and moments acting between bodies can be displayed by attaching further glyphs to the bodies. Instead of complex glyphs capable of displaying many different attributes, we opted for simpler ones, and allow several instances of them to be created to visualize different attributes. We found that this approach is more flexible in the context of the chain and belt drive analysis.

A *colored disk* is a very simple glyph that can represent a scalar value, a component of a vector, or a vector's magnitude by mapping it to a color in a color gradient. Displaying connection forces by showing colored disks at each chain link provides a simple way of locating areas of extreme forces, which is often the first step when investigating chain drives. We offer three approaches to map scalars to colors. The color gradient can be mapped to: (1) the minimum and maximum of the selected data attribute over all simulation time steps, (2) the minimum and maximum in the current time step, or (3) a user defined range. These choices allow the user to balance consistent coloring during animation versus accessing details in the current time step, and also to "zoom in" on a specific range of result values.

Vector quantities are intuitively visualized by *arrows glyphs*. We offer several drawing styles for arrow glyphs, so that one can use different styles for the simultaneous visualization of different vector attributes and avoid confusion. The complete 3D vector or some of its components can be shown. It is also possible to show any component along *any* axis of the local coordinate system. This is useful when the tangential velocities of chain links is displayed *normal* to the chain drive, in order to avoid occlusion.

Angular velocity and acceleration can be visualized by the *sector glyph*. The sector's center is at the center of gravity of the body. The normal of the sector coincides with axis of rotation. The magnitude is visualized by the angle of the sector. The direction of the rotation (clockwise vs. counter-clockwise) is indicated by changing the transparency along the arc.

A disadvantage of the sector glyph is that it is flat and when viewed under very small angles it is difficult to interpret. In order to reliably visualize angular attributes independent from the view direction, we introduce the *spiral glyph*. A star is twisted around its center, and the resulting spiral is mapped onto the surface of a cone. The amount of twist and the cone's height are functions of the attribute being visualized. The axis of the cone indicates the axis of rotation.

The mapping from physical attributes to geometric properties (arrow length, sector angle, spiral height and twist) follows the principles described above for coloring the disk glyphs. Annotations displaying actual numeric values can be attached to the glyphs to support quantitative analysis.

Elastic Multibody Dynamics

Many components of internal combustion engines suffer significant deformations when the engine is running. Conrods bend under load during combustion, crankshafts endure bending and torsional deformation, and the engine block itself vibrates as well. The deformation can be simulated based on the theory of *elastic multibody systems*, where each body models a structural engine component.

The visual representation of a body includes a *finite element (FE) mesh*, a simplified beam mesh on which the simulator operates, and several *glyphs*. Occlusion is a major issue, since virtually everything (crankshaft, conrods, etc.) that is of interest to the engineer is contained in the engine block. We offer four means of *reducing occlusion* in FE meshes: (1) the outermost layers of FE meshes can be peeled, (2) clipping planes can be applied to each FE mesh, (3) an intersection plane can be used to show only elements that intersect the plane, and (4) arbitrary groups of FE elements can be selected and hidden. Transparency can be combined with any of the above tools. Before simulation, the engineer can define constraints on certain degrees of freedom of specific nodes. These constraints are visualized by glyphs attached to the nodes. The local coordinate system and the axis of rotation (when applicable) are also indicated by glyphs.

The simulation computes the global motion of the body and the local displacement of each node. The deformation of bodies can be visualized by directly displacing their nodes. The deformation computed in simulation usually needs to be *magnified* to become visible. The *torsional deformation* of rotating components (e.g., crankshafts) is often of special interest. Simple linear magnification of the displacement can distort parts of the body in strange, disproportional ways, and the torsional deformation is very difficult to discern. The situation can be improved by magnifying the displacements in a cylindrical coordinate system defined by the body's center of gravity and axis of rotation. If the scaling factors of the radial and axial components are set to zero and the angular scaling factor is nonzero, then only the angular component of the deformation is preserved. The resulting visualization better depicts the torsional deformation.

Attributes can be displayed using color mapping on the *surface* of the FE mesh or on arbitrary *cross-sections* across the mesh. The mapping of physical attributes to color follows the previously described principles for coloring glyphs. The previously introduced arrow, sector and spiral glyphs can be attached to the mesh nodes to visualize results. Attaching glyphs to each node of a high resolution mesh can create busy, cluttered, and uninformative visualizations. We adapted the *resampling* tool proposed by Laramee [145]. Glyphs can be positioned in the reduced resolution resampling grid in order to improve the visual quality.

8.4 Application Examples

The main motivation behind this thesis was to improve the visual exploration and analysis of data sets in automotive engineering. In this section we summarize two case studies we did together with engineers working at AVL-List GmbH (`www.avl.com`), a company active in internal combustion engine research and development. Furthermore, we mention several application areas outside engineering where the methodology described in this thesis has been used successfully.

Diesel Fuel Injection System Simulation

There are many (often conflicting) goals of Diesel engine design including high power, fuel efficiency, reducing noise levels, and meeting emission regulations. The fuel injection system is the key engine component to achieve those goals, and its behavior can be simulated using 1D computational fluid dynamics. We analyzed a common rail Diesel fuel injection system [29].

Over 4000 simulations of the injection process were computed, with different input fuel pressure and injection timing setting used as control parameters. For each combination of the control parameters, the simulation computes the amount of fuel injected as a function of time. Together they constitute a family of function graphs. Similarly, the injection pressure and the position of the needle in the injector are also given as families of function graphs. The ideal shape of the injected fuel amount function graphs is different for different operating conditions (engine speed and load). The goal is to identify the optimal combinations of control parameters that produce the desired shapes. Those parameters are programmed into the ECU (Electronic Control Unit) that manages the engine.

During the analysis, the control parameters were displayed in scatter plots, histograms, and parallel coordinates. We could select the desired injection shapes using combinations of line brushes in function graph views, and observe the highlighted control parameters. We also learned, that when pressure on the injector's inlet increases too fast, then the resulting wave can be reflected into the fuel line, which impairs our control over the injection shape. We also found that the injection rate and the needle lift function graphs are tightly correlated.

Timing Chain Drive

Timing chains transfer the rotation of the engine's crankshaft to the camshaft which in turn actuates the valves, synchronized to the piston's motion. Especially at high engine speeds, dynamic and inertial phenomena cause vibrations that increase noise levels and mechanical wear, and deteriorate the accuracy of the motion coupling. The chain's motion can be computed based on rigid body dynamics when each link, as well as the sprocket and guides in the system are considered rigid bodies [241].

More than 1100 simulations were computed with different control parameter settings (stiffness of the sprockets and guides, preload force on the chain) and at different engine speeds. The simulation results include function graphs that describe the maximum connection force between chain links over the simulation time. The independent variable of the function graphs is not time, but rather the *index* of the link in the chain. The Fourier transform of the camshaft sprocket's rotational velocity is also given as a family of function graphs.

We used the *segmented curve view*, since the domain of the function graphs (chain link index) is discrete. During the analysis, we identified an invalid parameter combination which produced extremely high contact forces. This was also confirmed in the 3D view: the chain vibrated wildly. We learned that the connection forces between chain links depend primarily on the sprocket's offset and, to a lesser extent, on the chain's preload. The contact forces depend on the sprocket and guide stiffness settings. After understanding the dependencies between parameters and forces, in an iterative optimization process we could find a combination of parameters that represents a favorable compromise in reducing noise and maximizing durability.

Application Examples outside Engineering

The framework presented in this thesis is suitable also for the analysis of data from problem domains outside engineering. We have successfully used the same principles to analyze the *evacuation* of a building [74], a *social network* [137], and a *geospatial-temporal* data set [173]. Based on a collaboration with medical experts, the system has been adapted and improved to support the analysis of *medical measurement data* recorded at the intensive care unit of a hospital [160]. Another enhancement of the system has been used, in cooperation with ethologists, to analyze animals' paths of motion in behavioral experiments [168].

8.5 Discussion

Sophisticated tools are required to explore and analyze complex, high-dimensional, and time-dependent data sets generated by the simulation of car components and subsystems. Computational analysis methods have been widely used for such data sets. This thesis describes different, interactive visual analysis based approaches, which allow users to interact with the data and aid engineers in getting insight and generate useful knowledge from the raw numbers in a guided human-computer dialogue. We described a framework based on the concept of coordinated multiple views, specialized visualizations for time-dependent data, integrated computation of derived data attributes, and interaction techniques that support interactive data analysis. We also introduced novel 3D visualizations for multibody systems. The selection of tools and methods presented support common tasks, such as parameter sensitivity analysis and optimization. We discussed the benefits of those techniques and how engineers gain valuable insight into real-world simulation data using the proposed set of tools.

Chapter 9

Conclusions

"Everything's got a moral if only you can find it."

— Lewis Carroll (1832–1898)[1]

Simulations of complex systems are routinely performed in several applications domains, including automotive engineering and meteorology. The simulation can be repeated with perturbed input parameters to represent different boundary conditions or design choices. The goal of the analysts is to generate useful knowledge from the complex simulation results. Interactive visual analysis is seen as a valuable approach to assist them in this process [88]. Simulation results are multi-run, multivariate, and time-dependent. The analysis of this kind of data is particularly challenging, because analysis tools need to address all of those different data characteristics [118]. This thesis presented interactive visual tools and methods for the analysis of such complex data sets. Although our work is motivated by the need for the analysis of simulation results in the automotive industry, the discussed concepts are applicable to several other problem domains.

We use coordinated multiple views [214] to display different perspectives of the data. Novel visualizations for time-dependent data are integrated. A rich set of interaction possibilities enables the user to focus on interesting features. Focus+context visualization helps relate features to the rest of the data [59]. The focus can be iteratively changed, refined, or broadened, reflecting the user's mental process during exploration. The rapid brushing and studying of the linked views helps in the development of a mental model of the relations in the data. Correlations in multivariate results can be found, and relationships between features in parameter space and in result space can be explored.

We have integrated the on-demand computation of derived data attributes [61], such as extrema and first derivative of time-dependent data, into the visual analysis. This enables the analyst to select also those features that cannot be specified via simple brushing of the original data variates. The combination of those building blocks allows the users to explore and analyze their data effectively. They can formulate and evaluate hypotheses visually to generate insight. They can use the insight gained to refine their hypotheses or generate new ones, making the process

[1] English author, mathematician, and logician. His most notable work is *Alice's Adventures in Wonderland.*

iterative. They can achieve complex analysis goals, such as exploring the system's sensitivity to its parameters and finding optimal simulation parameter values. We also propose 3D visualizations to capture the spatio-temporal aspect of data generated by simulations of multibody dynamics.

Much of the work presented in this thesis was done while collaborating with engineers working in automotive engine design. This collaboration was very inspiring, because we had a chance to understand what they want to achieve, what they expect from the visualization, and how they make sense of data. It was interesting to observe that engineers and the visualization community often have very different views on the analysis, even on basic terminology. Indeed, the two groups sometimes have markedly different concepts of "data", "visualization", and "analysis".

We found that 3D visualizations were generally well-received. Engineers deemed them intuitive and they were able and willing to work productively with them. On the contrary, the more abstract visualizations integrated in coordinated multiple views were initially received with less enthusiasm. Interaction is crucial in working with such systems, and some of the engineers we collaborated with needed time to become able to leverage brushing and the coordination of views effectively (compare to Liu and Stasko [153]). Initially, they did not naturally engage in interactive exploration, but rather opted to use visualization only for confirmation. This is also reflected, to some extent, in their terminology, referring to visualization as "post-processing", with little interaction implied. The flexibility in interaction that a general purpose system should offer is perhaps too much when the user focuses on more specific tasks. For visual analysis solutions targeted at specific tasks and domains, it may be more productive to streamline the interface for ease of use, also considering domain knowledge in the design [109].

In the traditional workflow, the analysis of simulation data begins after the simulation has finished and all data is available. Recently, there is an increased interest in real-time analysis and also real-time, interactive steering of simulations [210]. A challenge for future work is to incorporate new data into the analysis progressively as it is being generated, without causing the users to lose their overview and focus.

It is generally accepted in the visualization community that visualization for presentation requires very different values compared to visualization for exploration and analysis. Still, most of the engineers we worked with wished to use the same tools to analyze data and to document and present their findings. Considering their workflow, this requirement sounds natural. Nevertheless, it is very difficult to provide tools that blend the interactivity, flexibility and performance required for exploration and analysis with the aesthetic values one expects in presentation. As a matter of fact, analysts often have to resort to using different, independent tools for the different purposes, which hinders their workflow [223]. We speculate that the better integration of visual analysis systems with the tools used for presentation could lead to their more wide-spread acceptance.

Acknowledgments

"The only thing that will redeem mankind is cooperation."

— Bertrand Russell (1872–1970)[1]

Completing this thesis would not have been possible without the support and cooperation from a lot people. First and foremost, I would like to thank my supervisor *Helwig Hauser* (now with the University of Bergen, formerly with the VRVis Research Center) for his support. He was always there to point out the most interesting directions of work and also to provide the most valuable constructive comments and feedback, exactly at the right time. Very special thanks go to *Krešimir Matković* (VRVis) for the immense professional and moral support. Thank you for steadily encouraging me to continue working on and eventually finishing this thesis.

This thesis was written while working for the *VRVis Research Center* (`www.vrvis.at`) and collaborating with *AVL-List GmbH* (`www.avl.com`), active in internal combustion engine research and development. AVL's contribution goes far beyond providing data sets for testing and demonstrating ideas. People at AVL contributed in many different ways to this thesis. They identified open issues and problems in their everyday work where interesting opportunities for interactive visual analysis could be found. Their input as domain experts is invaluable. I am thankful to my coauthors at AVL, in alphabetical order: *Mario Duras*, *Mario Jelović*, *Josip Jurić*, and *Jürgen Krasser*, as well as *Alan Lež* at VRVis. Furthermore, the long and fruitful discussions with *Saša Bukovnik*, *Wolfgang Hellinger*, *Bernhard Loibnegger*, *Günter Offner*, *Martin Sopouch*, and *Christian Vock* at AVL have made me more aware of the engineering background, which inevitably also made me understand the visualization and visual analysis problem better. Additional thanks go to *Audrey Cahill* (AVL), *Mark A. Mitterdorfer* (now with Intel Corporation, formerly with AVL), *Robert S. Laramee* (now with Swansea University, formerly with VRVis), *Johannes Tax*, and *Michaela Haller* for their feedback on my papers. I am also grateful to the anonymous reviewers for their constructive comments.

I am particularly thankful to *Denis Gračanin* (Virginia Tech) for the perfectly timed collaborations on papers and for using the time zone difference between Austria and the United States to our maximum advantage. He really knows how to make that productive. Special thanks go to *Meister Eduard Gröller* and *Werner Purgathofer* (Vienna University of Technology) for turning seemingly big problems into non-problems. This thesis could not have been finished without their support. I am grateful to *László Szirmay-Kalos* (Budapest University of Technology), the

[1] British philosopher, logician, mathematician, and social critic. Received the 1950 Nobel Prize in Literature in recognition of his varied and significant writings in which he champions humanitarian ideals and freedom of thought.

supervisor of my master's thesis. I have always been interested in computer graphics since I was about 12, but he was the one who encouraged me to write my first conference paper and started me out on the path of computer graphics research. After so many years, thank you.

Finally, but most importantly, thanks go to my lovely wife *Edit* who supported and encouraged me during the long days and nights before submitting papers and endured my occasional ramblings about visual analysis at home. Very special thanks and warm hugs to my daughters *Anna* and *Julia*, who missed Daddy so much while he was working on this thesis. I appreciate the sacrifices they made.

Curriculum Vitae

About Zoltán Konyha

Zoltán Konyha, MSc, born on 27 December 1976, in Győr, Hungary, as the first son of Dipl.-Ing. Márta Csifféry and Dipl. Ing.-László Konyha.
Married with Edit Csapó, MSc.
Two daughters, Anna (2006) and Júlia (2011).

Contact information

Address: Lilienthalgasse 39, A-8080 Graz, Austria. Email: konyha@vrvis.at

Education

1991–1995 **Révai Miklós Gimnázium** (high school) in Győr, Hungary. Graduation (Matura) with highest distinction in June 1995.

1995–2001 **Studies of Computer Science** at the Budapest University of Technology and Economics, focus on Business Telecommunication and Computer Aided Design of Information Systems.

2001 **Diploma thesis**: *Automobile Simulation in Virtual Environment* (in Hungarian). Supervisor: Dr. László Szirmay-Kalos. **Graduation to "Master of Science in Technical Informatics"**.

2006– **Phd student** in Computer Science (Informatik) at the Vienna University of Technology, resulting in work on this **Phd thesis** at the VRVis Research Center.

Professional activities

1998–1999 **Teaching assistant**, lab on the Unix operating system, at the Dennis Gabor College, Budapest, Hungary. (ref: Dr. Pál Jedlovszky)

2001–2012 **Junior Researcher** and Software Developer at VRVis Research Center, Vienna, Austria. (ref: Dipl.-Ing. Georg Stonawski)

2012– **Software Engineer** at AVL-List GmbH, Graz, Austria. (ref: Dr. Jürgen Krasser)

Activities related to scientific work

- **Research project** *Simulation and analysis of TCP traffic over ATM telecommunication networks* with Ericsson Research Hungary in 1998 during my studies at the Budapest University of Technology and Economics. (ref: Dr. Gergely Seres, Ericsson Research Hungary)
- **Diploma thesis**: *Automobile Simulation in Virtual Environment* (in Hungarian) at the Department of Control Engineering and Information Technology, Budapest University of Technology and Economics. Supervisor: Dr. László Szirmay-Kalos. (January 2001)
- **Reviewer for conferences**: *Advanced Visual Interfaces* (2010), *Eurographics / IEEE Symposium on Visualization* (2009-2010), *Spring Conference on Computer Graphics* (2010), *Winter School of Computer Graphics* (2010),
- **Co-author** of the "Simple and Effective Integrated Display" **award winning entry** [173] to the *VAST Challenge 2008* (a participation category of the *IEEE VAST 2008 Symposium*, part of *VisWeek 2008*).

Publications

Please see the **Related Publications** and the **Bibliography** sections of this thesis for a list of publications related to the contents of this thesis.

Bibliography

[1] S. Afzal, R. Maciejewski, and D. S. Ebert. Visual analytics decision support environment for epidemic modeling and response evaluation. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST 2011)*, pages 191–200, 2011.

[2] C. Ahlberg. Spotfire: An information exploration environment. *SIGMOD Record*, 25(4):25–29, 1996.

[3] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing time-oriented data – A systematic view. *Computer and Graphics*, 31(3):401–409, 2007.

[4] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47–60, 2008.

[5] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Human Computer Interaction. Springer Verlag, 1st edition, 2011.

[6] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization*, pages 115–122, 2007.

[7] O. S. Alabi, X. Wu, J. M. Harter, M. Phadke, L. Pinto, H. Petersen, S. Bass, M. Keifer, S. Zhong, C. G. Healey, and R. M. Taylor II. Comparative visualization of ensembles using ensemble surface slicing. In *Proceedings of Visualization and Data Analytics (VDA 2012)*, pages 1–12, 2012.

[8] B. Alpern and L. Carter. The hyperbox. In *Proceedings of the 2nd conference on Visualization (VIS '91)*, pages 133–139, 1991.

[9] G. Andrienko and N. Andrienko. Visual exploration of the spatial distribution of temporal behaviors. In *Proceedings of the Ninth International Conference on Information Visualisation (IV'05)*, pages 799–806, 2005.

[10] N. Andrienko and G. Andrienko. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[11] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):12–21, 1973.

[12] A. Aroussi, A. Hassan, and Y. Morsi. Numerical simulation of the airflow over and heat transfer through a vehicle windshield defrosting and demisting system. *Heat and Mass Transfer*, 39(5):401–405, 2003.

[13] D. Asimov. The grand tour: A tool for viewing multidimensional data. *SIAM Journal on Scientific and Statistical Computing*, 6(1):128–143, 1985.

[14] P. Bak, F. Mansmann, H. Janetzko, and D. Keim. Spatiotemporal analysis of sensor logs using growth ring maps. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):913–920, 2009.

[15] M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the working conference on Advanced Visual Interfaces (AVI '00)*, pages 110–119, 2000.

[16] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3):223–232, 1989.

[17] T. R. Barrett, N. W. Bressloff, and A. J. Keane. Airfoil design and optimization using multi-fidelity analysis and embedded inverse design. In *Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1–21, 2006.

[18] O. A. Bauchau. Computational schemes for flexible, nonlinear multi-body systems. *Multibody System Dynamics*, 2:169–225, 1998.

[19] E. Bauer and A. Ganopolski. Simulation of the cold climate event 8200 years ago by meltwater outburst from Lake Agassiz. *Paleoceanography*, 19(PA3014):1–13, 2004.

[20] C. Baumgarten. *Mixture Formation in Internal Combustion Engines*. Springer, 2006.

[21] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[22] W. Berger, H. Piringer, P. Filzmoser, and M. E. Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum*, 30(3):911–920, 2011.

[23] J. Bertin. *Semiology of graphics*. University of Wisconsin Press, 1983.

[24] E. Bertini, L. Dell' Aquila, and G. Santucci. Springview: Cooperation of radviz and parallel coordinates for view optimization and clutter reduction. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '05)*, pages 22–29, 2005.

[25] E. Bertini and D. Lalanne. Surveying the complementary role of automatic data analysis and visualization in knowledge discovery. In *Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery: Integrating Automated Analysis with Interactive Exploration (VAKD '09)*, pages 12–20, New York, NY, USA, 2009. ACM.

[26] K. Binder and D. W. Heermann. *Monte Carlo Simulation in Statistical Physics: An Introduction*, volume 80 of *Springer Series in Solid-State Sciences*. Springer, 2010.

[27] A. R. Binesh and S. Hossainpour. Three dimensional modeling of mixture formation and combustion in a direct injection heavy-duty Diesel engine. *World Academy of Science, Engineering and Technology*, 17:207–212, 2008.

[28] J. Blaas, C. P. Botha, and F. H. Post. Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1436–1451, 2008.

[29] W. Boehner and K. Hummel. Common Rail Injection System for Commercial Diesel Vehicles. *SAE Transactions, SAE 970345*, 1997.

[30] D. Borland and R. M. Taylor II. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, 27(2):14–17, 2007.

[31] K. Brodlie, R. AllendesOsorio, and A. Lopes. A review of uncertainty in data visualization. In J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P. C. Wong, editors, *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109. Springer London, 2012.

[32] A. Buja and D. Asimov. Grand tour methods: An outline. In *Proceedings of the Seventeenth Symposium on the Interface of Computer Sciences and Statistics*, pages 63–67, 1986.

[33] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *Proceedings of the 2nd conference on Visualization (VIS '91)*, pages 156–163, 1991.

[34] P. Buono, A. Aris, C. Plaisant, A. Khella, and B. Shneiderman. Interactive pattern search in time series. In *Proceedings of Visualization and Data Analysis (VDA 2005)*, pages 175–186, 2005.

[35] P. Buono and A. L. Simeone. Interactive shape specification for pattern search in time series. In *Proceedings of the working conference on Advanced visual interfaces (AVI '08)*, pages 480–481, 2008.

[36] R. Bürger and H. Hauser. Visualization of multi-variate scientific data. In *EuroGraphics 2007 State of the Art Reports*, pages 117–134, 2007.

[37] R. Bürger, P. Muigg, H. Doleisch, and H. Hauser. Interactive cross-detector analysis of vortical flow data. In *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '07)*, pages 98–110, 2007.

[38] R. Bürger, P. Muigg, M. Ilcik, H. Doleisch, and H. Hauser. Integrating local feature detectors in the interactive visual analysis of flow simulation data. In K. Museth, T. Möller, and A. Ynnerman, editors, *Proceedings of Eurographics/ IEEE-VGTC Symposium on Visualization 2007*, pages 171–178, 2007.

[39] L. Byron and M. Wattenberg. Stacked graphs — geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008.

[40] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., 1999.

[41] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology (CHI '91)*, pages 181–186, New York, NY, USA, 1991. ACM.

[42] J. V. Carlis and J. A. Konstan. Interactive visualization of serial periodic data. In *Proceedings of the 11th annual ACM Symposium on User Interface Software and Technology (UIST '98)*, pages 29–38, 1998.

[43] D. B. Carr. Looking at large data sets using binned data plots. In A. Buja and P. A. Tukey, editors, *Computing and Graphics in Statistics*, pages 7–39. Springer-Verlag New York, Inc., 1991.

[44] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Chapman and Hall, New York, 1983.

[45] Y.-H. Chan, C. D. Correa, and K.-L. Ma. Flow-based scatterplots for sensitivity analysis. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST 2010)*, pages 43–50, 2010.

[46] P. Chaufour, G. Millet, S. Neyrat, M. Hedna, and E. Botelle. Advanced modeling of a heavy-truck unit-injector system and its applications in the engine design process. In *SAE 2004 World Congress & Exhibition*, 2004.

[47] C. Chen. An information-theoretic view of visual analytics. *IEEE Computer Graphics and Applications*, 28(1):18–23, 2008.

[48] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68(342):361–368, 1973.

[49] W. S. Cleveland. *The Elements of Graphing Data*. Wadsworth Publ. Co., Belmont, CA, USA, 1985.

[50] A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):2:1–2:31, 2009.

[51] G. Convertino, J. Chen, B. Yost, Y.-S. Ryu, and C. North. Exploring context switching and cognition in dual-view coordinated visualizations. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '03)*, pages 55–62, 2003.

[52] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, 1994.

[53] O. Daae Lampe and H. Hauser. Curve density estimates. *Computer Graphics Forum*, 30(3):633–642, 2011.

[54] W. C. de Leeuw and J. J. van Wijk. A probe for local flow field visualization. In *Proceedings Visualization '93*, pages 39–45, 1993.

[55] M. C. F. de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394, 2003.

[56] A. de Risi, P. Carlucci, T. Donateo, and A. Ficarella. A combined optimization method for common rail Diesel engines. In *ASME 2002 Proceedings of the Internal Combustion Engine Division Spring Technical Conference (ICES2002)*, pages 243–250, 2002.

[57] P. J. Diggle, P. Heagerty, K.-Y. Liang, and S. L. Zeger. *Analysis of Longitudinal Data*, volume 25 of *Oxford Statistical Science Series*. Oxford University Press, USA, 2002.

[58] H. Doleisch. SIMVIS: Interactive visual analysis of large and time-dependent 3D simulation data. In *Proceedings of the 39th conference on Winter Simulation Conference (WSC '07)*, pages 712–720, 2007.

[59] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the 5th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2003)*, pages 239–248, 2003.

[60] H. Doleisch and H. Hauser. Smooth brushing for focus+context visualization of simulation data in 3D. *Journal of WSCG*, 10(1):147–154, 2002.

[61] H. Doleisch, H. Hauser, M. Gasser, and R. Kosara. Interactive focus+context analysis of large, time-dependent flow simulation data. *Simulation*, 82(12):851–865, 2006.

[62] H. Doleisch, M. Mayer, M. Gasser, and H. Hauser. Case study: Visual analysis of complex, time-dependent simulation results of a Diesel exhaust system. In *Proceedings of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pages 91–96, 2004.

[63] H. Doleisch, M. Mayer, M. Gasser, P. Priesching, and H. Hauser. Interactive feature specification for simulation data on time-varying grids. In *Proceedings of the Conference on Simulation and Visualization 2005 (SimVis 2005)*, pages 291–304, 2005.

[64] S. P. Edwards, A. D. Pilley, S. Michon, and G. Fournier. The optimisation of common rail FIE equipped engines through the use of statistical experimental design, mathematical modeling and genetic algorithms. *SAE Transactions, SAE 97346*, 1997.

[65] D. C. Eleni, T. I. Athanasios, and M. P. Dionissios. Evaluation of the turbulence models for the simulation of the flow over a National Advisory Committee for Aeronautics (NACA) 0012 airfoil. *Journal of Mechanical Engineering Research*, 4(3):100–111, 2012.

[66] J. A. Fails, A. K. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology (VAST 2006)*, pages 167–174, 2006.

[67] Z. Fang, T. Möller, G. Hamarneh, and A. Celler. Visualization and exploration of time-varying medical image data sets. In *Proceedings of Graphics Interface 2007 (GI '07)*, pages 281–288, 2007.

[68] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of the 3rd annual ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST '90)*, pages 76–83, 1990.

[69] J.-D. Fekete, J. J. Wijk, J. T. Stasko, and C. North. The value of information visualization. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization*, pages 1–18. Springer-Verlag, 2008.

[70] R. Fellini, N. Michelena, P. Papalambros, and M. Sasena. Optimal design of automotive hybrid powertrain systems. In *Proceedings of the First International Conference on Environmentally Conscious Design and Inverse Manufacturing (ecodesign'99)*, pages 400–405, 1999.

[71] C. Fettes and A. Leipertz. Potentials of a piezo-driven passenger car common rail system to meet future emission legislations – an evaluation by means of in-cylinder analysis of injection and combustion. *SAE Technical Paper 2001-01-3499*, 2001.

[72] B. Francis and J. Pritchard. Visualisation of historical events using Lexis pencils. In D. Unwin and P. Fisher, editors, *Case studies of visualisation in the social sciences*. JISC Advisory group on Computer Graphics, Bristol, 1997.

[73] A. U. Frank. Different types of "times" in GIS. In M. J. Egenhofer and R. G. Golledge, editors, *Spatial and Temporal Reasoning in Geographic Information Systems*. Oxford University Press, New York, 1998.

[74] W. Freiler, K. Matković, Z. Konyha, D. Gračanin, T. Lipić, R. Miklin, and M. Berić. Analysis of evacuation traces. In *IEEE VisWeek Compendium*, pages 135–136, 2008.

[75] M. Friendly. Mosaic displays for multi-way contingency tables. *Journal of the American Statistical Association*, 89(425):190–200, 1994.

[76] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the conference on Visualization (VIS '99)*, pages 43–50, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[77] R. Fuchs and H. Hauser. Visualization of multi-variate scientific data. *Computer Graphics Forum*, 28(6):1670–1690, 2009.

[78] R. Fuchs, J. Waser, and M. E. Gröller. Visual human+machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1327–1334, 2009.

[79] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*, pages 16–23, 1986.

[80] G. W. Furnas and A. Buja. Prosection views: Dimensional inference through sections and projections. *Journal of Computational and Graphical Statistics*, 3(4):323–353, 1994.

[81] V. González and A. Kobsa. A workplace study of the adoption of information visualization systems. In *3rd International Conference on Knowledge Management (I-KNOW'03)*, pages 92–102, 2003.

[82] G. Greeves, S. Tullis, and B. Barker. Advanced two-actuator EUI and emission reduction for heavy-duty Diesel engines. *SAE Transactions*, 112(3):914–931, 2003.

[83] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: A system for visually linking 3-D and statistical visualizations applied to cardiac simulation and measurement data. In *Proceedings of the IEEE Visualization (VIS 2000)*, pages 489–492, 2000.

[84] H. Griethe and H. Schumann. The visualization of uncertain data: Methods and problems. In *Proceedings of the Conference on Simulation and Visualization 2006 (SimVis 2006)*, pages 143–156, 2006.

[85] D. M. Hamby. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental Monitoring and Assessment*, 32(2):135–154, 1994.

[86] M. C. Hao, M. Marwah, H. Janetzko, U. Dayal, D. A. Keim, D. Patnaik, N. Ramakrishnan, and R. K. Sharma. Visual exploration of frequent patterns in multivariate time series. *Information Visualization*, 11(1):71–83, 2012.

[87] H. Hauser. Generalizing focus+context visualization. In *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Mathematics and Visualization, pages 305–327. Springer, 2005.

[88] H. Hauser. Interactive visual analysis - an opportunity for industrial simulation. In *Proceedings of the Conference on Simulation and Visualization 2006 (SimVis 2006)*, pages 1–6, 2006.

[89] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS '02)*, pages 127–130, 2002.

[90] S. Havre, E. G. Hetzler, P. Whitney, and L. T. Nowell. ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

[91] J. C. Helton and F. J. Davis. Illustration of sampling-based methods for uncertainty and sensitivity analysis. *Risk Analysis*, 22(3):591–622, 2002.

[92] J. C. Helton, J. D. Johnson, C. J. Sallaberry, and C. B. Storlie. Survey of sampling-based methods for uncertainty and sensitivity analysis. *Reliability Engineering & System Safety*, 91(10-11):1175–1209, 2006.

[93] C. Henze. Feature detection in linked derived spaces. In *Proceedings of the conference on Visualization (VIS '98)*, pages 87–94, 1998.

[94] J. L. Hintze and R. D. Nelson. Violin plots: A box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

[95] T. Hiroyasu, M. Miki, M. Kim, S. Watanabe, H. Hiroyasu, and H. Miao. Reduction of heavy duty Diesel engine emission and fuel economy with multi-objective genetic algorithm and phenomenological model. *SAE Technical Paper 2004-01-0531*, 2004.

[96] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[97] D. M. W. Hoffman and D. R. Dowling. Modeling fully coupled rigid engine dynamics and vibrations. In *SAE Noise & Vibrations Conference & Exposition*, 1999.

[98] P. Hoffman, G. Grinstein, and D. Pinkney. Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations. In *Workshop on New Paradigms in Information Visualization and Manipulation*, pages 9–16, 1999.

[99] P. E. Hoffman. *Table Visualizations: A Formal Model and its Applications*. PhD thesis, University of Massachusetts Lowell, 2000.

[100] D. Holten and J. J. van Wijk. Evaluation of cluster identification performance for different PCP variants. *Computer Graphics Forum*, 29(3):793–802, 2010.

[101] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[102] T. Jankun-Kelly and K. Mehta. Superellipsoid-based, real symmetric traceless tensor glyphs motivated by nematic liquid crystal alignment visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1197–1204, 2006.

[103] W. Javed, B. McDonnel, and N. Elmqvist. Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):927–934, 2010.

[104] D. H. Jeong, C. Ziemkiewicz, B. D. Fisher, W. Ribarsky, and R. Chang. iPCA: An interactive system for PCA-based visual analytics. *Computer Graphics Forum*, 28(3):767–774, 2009.

[105] D. F. Jerding and J. T. Stasko. The information mural: A technique for displaying and navigating large information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):257–271, 1998.

[106] J. Johansson, C. Forsell, M. Lind, and M. Cooper. Perceiving patterns in parallel coordinates: determining thresholds for identification of relationships. *Information Visualization*, 7(2):152–162, 2008.

[107] J. Johansson, P. Ljung, and M. Cooper. Depth cues and density in temporal parallel coordinates. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization (EUROVIS'07)*, pages 35–42, 2007.

[108] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing structure within clustered parallel coordinates displays. In *Proceedings of the 2005 IEEE Symposium on Information Visualization (INFOVIS '05)*, pages 125–132, 2005.

[109] C. R. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, 2004.

[110] C. R. Johnson and A. R. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 23(5):6–10, 2003.

[111] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[112] E. Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *Proceedings of the 7th ACM SIGKDD Intrnational Conference on Knowledge Discovery and Data mining*, pages 107–116, 2001.

[113] D. Kao, J. L. Dungan, and A. Pang. Visualizing 2d probability distributions from EOS satellite image-derived data sets: A case study. In *Proceedings of the conference on Visualization (VIS '01)*, pages 457–460, 2001.

[114] D. T. Kao, A. Luo, J. L. Dungan, and A. Pang. Visualizing spatially varying distribution data. In *Proceedings of the Sixth International Conference on Information Visualisation (IV '02)*, pages 219–226, 2002.

[115] T. Kapler and W. Wright. GeoTime information visualization. *Information Visualization*, 4(2):136–146, 2005.

[116] D. Keefe, M. Ewert, W. Ribarsky, and R. Chang. Interactive coordinated multiple-view visualization of biomechanical motion data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1383–1390, 2009.

[117] J. Kehrer, P. Filzmoser, and H. Hauser. Brushing moments in interactive visual analysis. *Computer Graphics Forum*, 29(3):813–822, 2010.

[118] J. Kehrer and H. Hauser. Visualization and visual analysis of multi-faceted scientific data: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 2012. Preprint, 10 April 2012.

[119] J. Kehrer, F. Ladstädter, P. Muigg, H. Doleisch, A. Steiner, and H. Hauser. Hypothesis generation in climate research with interactive visual data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1579–1586, 2008.

[120] J. Kehrer, P. Muigg, H. Doleisch, and H. Hauser. Interactive visual analysis of heterogeneous scientific data across an interface. *IEEE Transactions on Visualization and Computer Graphics*, 17(7):934–946, 2011.

[121] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization*, pages 154–175. Springer-Verlag, 2008.

[122] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.

[123] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[124] D. A. Keim, M. C. Hao, and U. Dayal. Hierarchical pixel bar charts. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):255–269, 2002.

[125] D. A. Keim, M. C. Hao, U. Dayal, and M. Hsu. Pixel bar charts: a visualization technique for very large multi-attribute data sets. *Information Visualization*, 1(1):20–34, 2002.

[126] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, editors. *Mastering The Information Age - Solving Problems with Visual Analytics*. Eurographics, 2010.

[127] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In S. J. Simoff, M. H. Böhlen, and A. Mazeika, editors, *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*, pages 76–90. Springer, 2008. Lecture Notes in Computer Science (LNCS).

[128] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Proceedings of the conference on Information Visualization (IV '06)*, pages 9–16, 2006.

[129] D. A. Keim, F. Mansmann, and J. Thomas. Visual analytics: How much visualization and how much analytics? *SIGKDD Explorations Journal*, 11(2):5–8, 2010.

[130] D. A. Keim, W. Müller, and H. Schumann. Visual data mining. In D. Fellner and R. Scopigno, editors, *STAR Proceedings of Eurographics 2002*. Eurographics Association, 2002.

[131] R. Keller, C. M. Eckert, and P. J. Clarkson. Multiple views to support engineering change management for complex products. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '05)*, pages 33–41, 2005.

[132] R. Kincaid and H. Lam. Line graph explorer: Scalable display of line graphs using focus+context. In *Proceedings of the working conference on Advanced Visual Interfaces (AVI '06)*, pages 404–411, 2006.

[133] G. Kindlmann. Superquadric tensor glyphs. In *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization (VISSYM'04)*, pages 147–154, 2004.

[134] T. K. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

[135] Z. Konyha, J. Jurić, K. Matković, and J. Krasser. Visualization of elastic body dynamics for automotive engine simulations. In *Proceedings of IASTED Visualization, Imaging, and Image Processing (VIIP 2004)*, pages 742–747, 2004.

[136] Z. Konyha, A. Lež, K. Matković, M. Jelović, and H. Hauser. Interactive Visual Analysis of Families of Curves using Data Aggregation and Derivation. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies (i-KNOW '12)*, pages 31–38, 2012.

[137] Z. Konyha, K. Matković, W. Freiler, R. Miklin, T. Lipić, M. Berić, and D. Gračanin. Analysis of cell phone calls history. In *IEEE VisWeek Compendium*, pages 133–134, 2008.

[138] Z. Konyha, K. Matković, D. Gračanin, and M. Duras. Interactive Visual Analysis of a Timing Chain Drive Using Segmented Curve View and other Coordinated Views. In *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '07)*, pages 3–15, 2007.

[139] Z. Konyha, K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive Visual Analysis of Families of Function Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1373–1385, 2006.

[140] Z. Konyha, K. Matković, and H. Hauser. Interactive 3D visualization of rigid body systems. In *Proceedings of IEEE Visualization 2003 (CIS 2003)*, pages 539–546, 2003.

[141] Z. Konyha, K. Matković, and H. Hauser. Interactive visual analysis in engineering: A survey. In *Poster Proceedingss of the 25th Spring Conference on Computer Graphics (SCCG 2009)*, pages 31–38, 2009.

[142] R. Kosara, F. Bendix, and H. Hauser. Timehistograms for large, time-dependent data. In *Proceedings of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pages 45–54, 2004.

[143] R. Kosara, G. N. Sahling, and H. Hauser. Linking scientific and information visualization with interactive 3D scatterplots. In *Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 133–140, 2004.

[144] M. Kostoglou, P. Housiada, and A. G Konstandopoulos. Multi-channel simulation of regeneration in honeycomb monolithic Diesel particulate filters. *Chemical Engineering Science*, 58(14):3273–3283, 2003.

[145] R. S. Laramee. FIRST: A flexible and interactive resampling tool for CFD simulation data. *Computers and Graphics*, 27:905–916, 2003.

[146] R. S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen. Visual analysis and exploration of fluid flow in a cooling jacket. In *Proceedings of the IEEE Visualization 2005*, pages 623–630, 2005.

[147] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In *Proceedings of the 1st conference on Visualization (VIS '90)*, pages 230–237, 1990.

[148] H. Levkowitz. Color icons: merging color and texture perception for integrated visualization of multiple parameters. In *Proceedings of the 2nd conference on Visualization (VIS '91)*, pages 164–170, 1991.

[149] H. Levkowitz. Perceptual steps along color scales. *International Journal of Imaging Systems and Technology*, 7:97–101, 1996.

[150] H. Levkowitz. *Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[151] H. Levkowitz and G. T. Herman. Color scales for image data. *IEEE Computer Graphics and Applications*, 12(1):72–80, 1992.

[152] J. Lin, E. Keogh, and S. Lonardi. Visualizing and discovering non-trivial patterns in large time series databases. *Information Visualization*, 4(2):61–82, 2005.

[153] Z. Liu and J. T. Stasko. Mental models, visual reasoning and interaction in information visualization: A top-down perspective. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):999–1008, 2010.

[154] B. Loibnegger, T. Resch, and B. Žmire. Enhanced Crankshaft Vibration and Stress Analysis. In *19th CAD-FEM Users' Meeting 2001*, 2001.

[155] K.-L. Ma. Machine learning to boost the next generation of visualization technology. *IEEE Computer Graphics and Applications*, 27(5):6–9, 2007.

[156] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[157] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, 2007.

[158] A. R. Martin and M. O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the 6th conference on Visualization (VIS '95)*, pages 271–278, 1995.

[159] K. Matković, W. Freiler, D. Gračanin, and H. Hauser. Comvis: A coordinated multiple views system for prototyping new visualization technology. In *Proceedings of the 2008 12th International Conference Information Visualisation (IV '08)*, pages 215–220, 2008.

[160] K. Matković, H. Gan, A. Ammer, D. Bennett, W. Purgathofer, and M. Terblanche. Interactive visual analysis of intensive care unit data: Relationship between serum sodium concentration, its rate of change, and survival outcome. In *Proceedings of the International Conference on Information Visualization Theory and Applications (IVAPP 2012)*, pages 648–659, 2012.

[161] K. Matković, D. Gračanin, M. Jelović, A. Ammer, A. Lež, and H. Hauser. Interactive visual analysis of multiple simulation runs using the simulation model view: Understanding and tuning of an electronic unit injector. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1449–1457, 2010.

[162] K. Matković, D. Gračanin, M. Jelović, and Y. Cao. Adaptive interactive multi-resolution computational steering for complex engineering systems. In *Proceedings of the International Workshop on Visual Analytics (EuroVA 2011)*, pages 45–48, 2011.

[163] K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual steering - rapid visual prototyping of a common rail injection system. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1699–1706, 2008.

[164] K. Matković, D. Gračanin, B. Klarin, and H. Hauser. Interactive visual analysis of complex scientific data as families of data surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1351–1358, 2009.

[165] K. Matković, D. Gračanin, Z. Konyha, and H. Hauser. Color lines view: An approach to visualization of families of function graphs. In *Proceedings of International Conference on Information Visualisation (IV '07)*, pages 59–64, Los Alamitos, CA, USA, 2007.

[166] K. Matković, M. Jelović, J. Jurić, Z. Konyha, and D. Gračanin. Interactive visual analysis and exploration of injection systems simulations. In *Proceedings of the IEEE Visualization (VIS 2005)*, pages 391–398, 2005.

[167] K. Matković, J. Jurić, Z. Konyha, J. Krasser, and H. Hauser. Interactive visual analysis of multiparameter families of function graphs. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV'05)*, pages 54–62, 2005.

[168] K. Matković, C. Winding, R. Splechtna, and M. Balka. Interactive visual analysis of ethological studies: Getting insight from large ensembles of animals' paths. In *International Workshop on Visual Analytics (EuroVA 2012)*, pages 85–89, 2012.

[169] B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in scientific computing. *Computer Graphics*, 21(6), 1987.

[170] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[171] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. Liverac: Interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 1483–1492, 2008.

[172] T. Mihalisin, J. Timlin, and J. Schwegler. Visualizing multivariate functions, data, and distributions. *IEEE Computer Graphics and Applications*, 11(3):28–35, 1991.

[173] R. Miklin, T. Lipić, Z. Konyha, M. Berić, W. Freiler, K. Matković, and D. Gračanin. Simple and effective integrated display: Geo-temporal analysis of migrant boats. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology, (IEEE VAST 2008)*, pages 203–204, 2008.

[174] R. B. Miller. Response time in man-computer conversational transactions. In *Proceedings of the Fall Joint Computer Conference, Vol 33, part I (AFIPS '68)*, pages 267–277, 1968.

[175] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 181–188, 1995.

[176] D. T. Montgomery and R. D. Reitz. Optimization of heavy-duty Diesel engine operating parameters using a response surface method. *SAE Technical Paper 2000-01-1962*, 2000.

[177] C. Z. Mooney. *Monte Carlo Simulation*, volume 116 of *Quantitative Applications in the Social Sciences*. Sage Publications, Incorporated, 1997.

[178] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser. A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data. *Computer Graphics Forum*, 27(3):775–782, 2008.

[179] W. Müller and H. Schumann. Visualization methods for time-dependent data - an overview. In *Proceedings of the 35th conference on Winter simulation (WSC '03)*, pages 737–745, 2003.

[180] G. Niemann and H. Winter. *Maschinenelemente*. Springer-Verlag, 1986.

[181] T. Nocke, M. Flechsig, and U. Böhm. Visual exploration and evaluation of climate-related simulation data. In *Proceedings of the 39th conference on Winter simulation (WSC '07)*, pages 703–711, 2007.

[182] D. A. Norman. Twelve issues for cognitive science. *Cognitive Science*, 4(1):1–32, 1980.

[183] C. North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26(3):6–9, 2006.

[184] C. North and B. Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *Proceedings of the working conference on Advanced visual interfaces (AVI '00)*, pages 128–135, 2000.

[185] M. Novotný and H. Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.

[186] L. Nowell, R. Schulman, and D. Hix. Graphical encoding for information visualization: An empirical study. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, pages 43–50, 2002.

[187] S. Oeltze, H. Doleisch, H. Hauser, P. Muigg, and B. Preim. Interactive visual analysis of perfusion data. *IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG)*, 13(6):1392–1399, 2007.

[188] G. Offner and H.-H. Priebsch. Flexible multi-body dynamics simulation - a powerful method for prediction of structure borne noise of internal combustion engines. In *Proceedings of the International Conference on Noise and Vibration Engineering (ISMA2006)*, pages 2663–2677, 2006.

[189] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997.

[190] R. Peng. A method for visualizing multivariate time series data. *Journal of Statistical Software, Code Snippets*, 25(1):1–17, 2008.

[191] W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS '04)*, pages 89–96, 2004.

[192] R. M. Pickett and G. G. Grinstein. Iconographic displays for visualizing multidimensional data. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 514–519, 1988.

[193] Z. Ping, O. Guangyao, B. Lufeng, W. Zhaowen, and D. Peng. Study on solid-fluid coupled heat transfer simulation of cylinder head of high power density Diesel engine. In *Proceedings of the International Conference on Intelligent Computation Technology and Automation, Volume 03, (ICICTA '10)*, pages 542–544, 2010.

[194] H. Piringer, W. Berger, and J. Krasser. HyperMoVal: Interactive visual validation of regression models for real-time simulation. *Computer Graphics Forum*, 29(3):983–992, 2010.

[195] H. Piringer, R. Kosara, and H. Hauser. Interactive focus+context visualization with linked 2D/3D scatterplots. In *Proceedings of the Second International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV'04)*, pages 49–60, 2004.

[196] H. Piringer, S. Pajer, W. Berger, and H. Teichmann. Comparative visual analysis of 2d function ensembles. *Computer Graphics Forum*, 31(3):1195–1204, 2012.

[197] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A multi-threading architecture to support interactive visual exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, 2009.

[198] D. Popović, M. Janković, S. Magner, and A. R. Teel. Extremum seeking methods for optimization of variable cam timing engine operation. *IEEE Transactions on Control Systems Technology*, 14(3):398–407, 2006.

[199] F. H. Post, F. J. Post, T. V. Walsum, and D. Silver. Iconic techniques for feature visualization. In *Proceedings of the 6th conference on Visualization (VIS '95)*, pages 288–295, 1995.

[200] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[201] K. Potter. Methods for presenting statistical information: The box plot. *Visualization of Large and Unstructured Data Sets*, S-4:97–106, 2006.

[202] K. Potter, J. Kniss, R. Riesenfeld, and C. R. Johnson. Visualizing summary statistics and uncertainty. *Computer Graphics Forum*, 29(3):823–832, 2010.

[203] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops (ICDMW '09)*, pages 233–240, 2009.

[204] M. Pournazeri, A. Khajepour, and A. Fazeli. An efficient lift control technique in electro-hydraulic camless valvetrain using variable speed hydraulic pump. *SAE Int. J. Engines*, 4(1):1247–1259, 2011.

[205] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[206] C. D. Rakopoulos, D. C. Rakopoulos, E. G. Giakoumis, and D. C. Kyritsis. Validation and sensitivity analysis of a two zone Diesel engine model for combustion and emissions prediction. *Energy Conversion and Management*, 45(9):1471–1495, 2004.

[207] R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '94)*, pages 318–322, 1994.

[208] F. Reinders, F. H. Post, and H. J. Spoelder. Visualization of time-dependent data with feature tracking and event detection. *The Visual Computer*, 17:55–71, 2001.

[209] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: Creating customized visualizations of complex data. *Computer*, 27(7):57–64, 1994.

[210] H. Ribičić, J. Waser, R. Fuchs, G. Blöschl, and E. Gröller. Visual analysis and steering of flooding simulations. *IEEE Transactions on Visualization and Computer Graphics (accepted for publication)*, 2012.

[211] H. Ribičić, J. Waser, R. Gurbat, B. Sadransky, and M. E. Gröller. Sketching uncertainty into simulations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2255–2264, 2012.

[212] R. E. Roberson and R. Schwertassek. *Dynamics of Multibody Systems*. Springer, 1988.

[213] J. C. Roberts. Exploratory visualization with multiple linked views. In A. MacEachren, M.-J. Kraak, and J. Dykes, editors, *Exploring Geovisualization*. Amsterdam: Elseviers, 2004.

[214] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '07)*, pages 61–71, 2007.

[215] B. E. Rogowitz and A. D. Kalvin. The "Which Blair Project": A quick visual method for evaluating perceptual color maps. In *Proceedings of the conference on Visualization '01*, pages 183–190, 2001.

[216] T. Ropinski, S. Oeltze, and B. Preim. Survey of glyph-based visualization techniques for spatial multivariate medical data. *Computers and Graphics*, 35(2):392–401, 2011.

[217] T. Ropinski and B. Preim. Taxonomy and usage guidelines for glyph-based medical visualization. In *Simulation and Visualization 2008 (SimVis 2008)*, pages 121–138, 2008.

[218] S. F. Roth, P. Lucas, J. A. Senn, C. C. Gomberg, M. B. Burks, P. J. Stroffolino, J. A. Kolojejchick, and C. Dunmire. Visage: A user interface environment for exploring information. In *Proceedings of the IEEE Information Visualization*, pages 3–12, 1996.

[219] M. Rottmann, C. Menne, S. Pischinger, V. Luckhchoura, and N. Peters. Injection rate shaping investigations on a small bore DI Diesel engine. *SAE Technical Paper 2009-01-0850*, 2009.

[220] E. A. Rundensteiner, M. O. Ward, J. Yang, and P. R. Doshi. XmdvTool: visual interactive data exploration and trend discovery of high-dimensional data sets. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD '02)*, pages 631–631, 2002.

[221] T. Saito, H. N. Miyamura, M. Yamamoto, H. Saito, Y. Hoshiya, and T. Kaseda. Two-tone pseudo coloring: Compact visualization for one-dimensional data. In *Proceedings of the 2005 IEEE Symposium on Information Visualization INFOVIS '05*, pages 23–30, 2005.

[222] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1421–1430, 2010.

[223] P. Saraiya, C. North, V. Lam, and K. A. Duca. An insight-based longitudinal study of visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1511–1522, 2006.

[224] J. Sauer and E. Schömer. A constraint-based approach to rigid body dynamics for virtual reality applications. In *Proceedings of the ACM Symposium on Virtual Reality software and technology*, pages 153–162, 1998.

[225] T. Schaffner, M. Sopouch, and H.-H. Priebsch. Chain noise excitation in combustion engines numerical simulation and verification of different chain types. In *Proceedings of the International Conference on Noise and Vibration Engineering*, pages 975–990, 2004.

[226] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware Inc., 3rd edition, 2003.

[227] H.-J. Schulz, A. M. Uhrmacher, and H. Schumann. Visual analytics for stochastic simulation in cell biology. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies (i-KNOW '11)*, pages 48:1–48:8, 2011.

[228] H. Schumann and W. Müller. *Visualisierung — Grundlagen Und Allgemeine Methoden*. Springer, 2000. (in German).

[229] R. Schwertassek, S. von Dombrowski, and O. Wallrapp. Modal Representation of Stress in Flexible Multibody Simulation. *Nonlinear Dynamics*, 20(4):381–399, 1999.

[230] M. Sedlmair, W. Hintermaier, K. Stocker, T. Büring, and A. Butz. A dual-view visualization of in-car communication processes. In *Proceedings of the 2008 12th International Conference Information Visualisation (IV '08)*, pages 157–162, 2008.

[231] M. Sedlmair, P. Isenberg, D. Baur, and A. Butz. Information visualization evaluation in large companies: Challenges, experiences and recommendations. *Information Visualization*, 10(3):248–266, 2011.

[232] M. Sedlmair, M. D. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012.

[233] P. K. Senecal, D. T. Montgomery, and R. D. Reitz. A methodology for engine design using multi-dimensional modelling and genetic algorithms with validation through experiments. *International Journal of Engine Research*, 1(3):229–248, 2000.

[234] J. Seo and B. Shneiderman. A rank-by-feature framework for unsupervised multidimensional data exploration using low dimensional projections. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS '04)*, pages 65–72, 2004.

[235] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[236] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, 1996.

[237] B. Shneiderman. Inventing discovery tools: combining information visualization with data mining. *Information Visualization*, 1(1):5–12, 2002.

[238] H. Siirtola. Interaction with the reorderable matrix. In *Proceedings of the Third International Conference on Information Visualisation (IV '99)*, pages 272–277, 1999.

[239] H. Siirtola and E. Mäkinen. Constructing and reconstructing the reorderable matrix. *Information Visualization*, 4(1):32–48, 2005.

[240] S. F. Silva and T. Catarci. Visualization of linear time-oriented data: A survey. In *Proceedings of the First International Conference on Web Information Systems Engineering (WISE'00)*, pages 310–319, 2000.

[241] M. Sopouch, W. Hellinger, and H. H. Priebsch. Prediction of vibroacoustic excitation due to the timing chains of reciprocating engines. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 217(3):225–240, 2003.

[242] R. Spence and L. Tweedie. The attribute explorer: information synthesis via exploration. *Interacting with Computers*, 11(2):137–146, 1998.

[243] D. F. Swayne, D. Temple Lang, A. Buja, and D. Cook. GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43:423–444, 2003.

[244] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnor, and D. A. Keim. Combining automated analysis and visualization techniques for effective exploration of high-dimensional data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*, pages 59–66, 2009.

[245] M. Theus. Interactive data visualization using mondrian. *Journal of Statistical Software*, 7(11):1–9, 2002.

[246] J. J. Thomas and K. A. Cook, editors. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center, 2005.

[247] J. J. Thomas and K. A. Cook. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, 2006.

[248] E. Thomsen. *OLAP Solutions: Building Multidimensional Information Systems*. John Wiley & Sons, Inc., New York, 1997.

[249] C. Tominski, J. Abello, and H. Schumann. Axes-based visualizations with radial layouts. In *Proceedings of the 2004 ACM symposium on Applied computing (SAC '04)*, pages 1242–1247, 2004.

[250] C. Tominski, P. Schulze-Wollgast, and H. Schumann. 3D information visualization for time dependent data on maps. In *Proceedings of the Ninth International Conference on Information Visualisation (IV '05)*, pages 175–181, 2005.

[251] E. R. Tufte. *The Visual Display Of Quantitative Information*. Graphics Press, 1983.

[252] E. R. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT 06410, 1990.

[253] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Reading, MA, 1977.

[254] B. Tversky, J. B. Morrison, and M. Bétrancourt. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[255] L. Tweedie, B. Spence, H. Dawkes, and H. Su. The Influence Explorer. In *Conference companion on Human factors in computing systems CHI '95*, pages 129–130, 1995.

[256] L. Tweedie, B. Spence, D. Williams, and R. Bhogal. The Attribute Explorer. In *Conference companion on Human factors in computing systems (CHI '94)*, pages 435–436, 1994.

[257] L. Tweedie, R. Spence, H. Dawkes, and H. Su. Externalising abstract mathematical models. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '96)*, pages 406–412, 1996.

[258] A. Unger, S. Schulte, V. Klemann, and D. Dransch. A visual analysis concept for the validation of geoscientific simulation models. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2216–2225, 2012.

[259] A. Unger and H. Schumann. Visual support for the understanding of simulation processes. In *IEEE Pacific Visualization Symposium (PacificVis 2009)*, pages 57–64, 2009.

[260] J. J. van Wijk. Image based flow visualization. *ACM Transactions on Graphics*, 21(3):745–754, 2002.

[261] J. J. van Wijk. The value of visualization. In *Proceedings of the 16th IEEE Visualization Conference (VIS 2005)*, pages 79–86, 2005.

[262] J. J. van Wijk and R. van Liere. Hyperslice: Visualization of scalar functions of many variables. In *Proceedings of the 4th conference on Visualization (VIS '93)*, pages 119–125, 1993.

[263] J. J. van Wijk and E. R. van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of the 1999 IEEE Symposium on Information Visualization (INFOVIS '99)*, pages 4–9, 1999.

[264] I. F. Vega López, R. T. Snodgrass, and B. Moon. Spatiotemporal aggregate computation: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):271–286, 2005.

[265] T. D. Wang, C. Plaisant, B. Shneiderman, N. Spring, D. Roseman, G. Marchand, V. Mukherjee, and M. Smith. Temporal summaries: Supporting temporal categorical searching, aggregation and comparison. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1049–1056, 2009.

[266] M. O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proceedings of the conference on Visualization (VIS '94)*, pages 326–333, 1994.

[267] M. O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3/4):194–210, 2002.

[268] M. O. Ward. Multivariate data glyphs: Principles and practice. In C.-H. Chen, W. Härdle, and A. Unwin, editors, *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics, pages 179–198. Springer, 2008.

[269] M. O. Ward and Z. Guo. Visual exploration of time-series data with shape space projections. *Computer Graphics Forum*, 30(3):701–710, 2011.

[270] C. Ware. *Information Visualization*. Morgan Kaufmann Publishers Inc., second edition, 2004.

[271] J. Waser, R. Fuchs, Ribičić, B. Schindler, G. Blöschl, and M. E. Gröller. World lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):1458–1467, 2010.

[272] M. Wattenberg. Sketching a graph to query a time-series database. In *Extended Abstracts on Human Factors in Computing Systems (CHI EA '01)*, pages 381–382, 2001.

[273] M. Wattenberg and J. Kriss. Designing for social data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):549–557, 2006.

[274] C. Weaver. Building highly-coordinated visualizations in Improvise. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 159–166, 2004.

[275] C. Weaver. Visualizing coordination in situ. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'05)*, pages 165–172, 2005.

[276] C. Weaver. Conjunctive visual forms. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):929–936, 2009.

[277] C. Weaver. Cross-filtered views for multidimensional visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):192–204, 2010.

[278] G. H. Weber, O. Rübel, M.-Y. Huang, A. H. DePace, C. C. Fowlkes, S. V. E. Keränen, C. L. Luengo Hendriks, H. Hagen, D. W. Knowles, J. Malik, M. D. Biggin, and B. Hamann. Visual exploration of three-dimensional gene expression using physical views and linked abstract views. *IEEE Transactions on Computational Biology and Bioinformatics*, 6(2):296–309, 2009.

[279] M. Weber, M. Alexa, and W. Müller. Visualizing time-series on spirals. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, pages 7–14. IEEE Computer Society, 2001.

[280] R. Wegenkittl, H. Löffelmann, and E. Gröller. Visualizing the behavior of higher dimensional dynamical systems. In *Proceedings of the IEEE Visualization (VIS '97)*, pages 119–125, 1997.

[281] J. Wei, H. Yu, R. Grout, J. Chen, and K.-L. Ma. Visual analysis of particle behaviors to understand combustion simulations. *IEEE Computer Graphics and Applications*, 32(1):22–33, 2012.

[282] J. J. v. Wijk. Bridging the gaps. *IEEE Computer Graphics and Applications*, 26(6):6–9, 2006.

[283] A. T. Wilson and K. C. Potter. Toward visual analysis of ensemble data sets. In *Proceedings of the 2009 Workshop on Ultrascale Visualization (UltraVis '09)*, pages 48–53, 2009.

[284] P. C. Wong and R. D. Bergeron. 30 years of multidimensional multivariate visualization. In *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33, 1997.

[285] J. Woodring and H.-W. Shen. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):123–137, 2009.

[286] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *Proceedings of the IEEE Symposium on Information Visualization 2003 (InfoVis 2003)*, pages 105–112, 2003.

[287] J. Yang, M. O. Ward, E. A. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In *Joint Eurographics/IEEE TCVG Symposium on Visualization (VisSym '03)*, pages 19–28, 2003.

[288] J. S. Yi, Y. a. Kang, J. T. Stasko, and J. A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[289] J. S. Yi, Y.-a. Kang, J. T. Stasko, and J. A. Jacko. Understanding and characterizing insights: How do people gain insights using information visualization? In *Proceedings of the 2008 conference on BEyond time and errors (BELIV '08)*, pages 1–6, 2008.

[290] H. Younesy, T. Möller, and H. Carr. Visualization of time-varying volumetric data using differential time-histogram table. In *Proceedings of the Fourth Eurographics / IEEE VGTC Workshop on Volume Graphics 2005*, pages 21–29. Eurographics Association, 2005.

[291] X. Yuan, P. Guo, H. Xiao, H. Zhou, and H. Qu. Scattering points in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1001–1008, 2009.

[292] L. Zhang, A. Zhang, and M. Ramanathan. Fourier harmonic approach for visualizing temporal patterns of gene expression data. In *Proceedings of the 2nd IEEE Computer Society Bioinformatics Conference (CSB 2003)*, pages 137–147, 2003.

[293] H. Zhou, X. Yuan, H. Qu, W. Cui, and B. Chen. Visual clustering in parallel coordinates. *Computer Graphics Forum*, 27(3):1047–1054, 2008.